UNIVERSIDAD CARLOS III

BACHELOR'S DEGREE WORK

Smartwatch-Based User Monitoring for Human-Robot Interaction

Author:
Ignacio Guillermo MARTÍNEZ RINCÓN

Supervisor:
Dr. José Carlos CASTILLO MONTOYA

A bachelor's work
for the degree of Computer Science

June 17, 2019

"Its a dream, a fearful dream, life is"

Marcus Aurelius

UNIVERSIDAD CARLOS III

Abstract

Universidad Carlos III, Madrid
Department of Automation Systems and Robotics

Computer Science

Smartwatch-Based User Monitoring for Human-Robot Interaction

by Ignacio Guillermo MARTÍNEZ RINCÓN

In the era of modern technologies such as the one we live in, there are several medical alert systems present in hospitals and in households. However, most of these systems present a faulty physical design, an unfriendly interface with users, or they are too expensive for the service that they offer. In my bachelor's work, I explain how an efficient, low-cost design and implementation of a medical alert system can be achieved, and can be used for the everyday life. I focus mainly on three things: drastically reducing the cost compared to competitors in the field, improving the effectiveness of the device when it's needed, and adapting the software to be interoperable and available to practically any robot anywhere in the world.

En la era de las tecnologías modernas como en la que vivimos actualmente, existen muchos sistemas de alerta médica que se pueden encontrar tanto en hospitales como en las casas. Sin embargo, la mayoría de estos sistemas están afectados por un diseño físico incorrecto, una interfaz poco cómoda con los usuarios, o simplemente son demasiado caros para el servicio que ofrecen. En mi trabajo de fin de grado, explico cómo se puede desarrollar un sistema de alerta médico a través de un diseño e implementaciones eficientes y de bajo coste; para poder ser usado en la vida diaria. Me centro principalmente en tres cosas: reducir el coste del sistema drásticamente respecto a los competidores en la industria, mejorar la efectividad del dispositivo cuando se necesita, y adaptar el software para que éste sea interoperable y esté disponible para ser usado prácticamente en cualquier robot en cualquier lugar del mundo.

Keywords: Computer Science, Stress, Medicine, ROS, Robot Operating System, Medical Alert System, Heart-Rate Monitor

Acknowledgements

Acknowledgements to my project advisor Dr. José Carlos Castillo Montoya for his extensive help in the development of my bachelor's work.

Also, to Elena Velázquez Navarro and Sergio González Díaz for helping me solve technical issues during this period.

Finally to my family, to whom I owe my success; for helping me to safely and successfully reach the finish line in this chapter of my life ...

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

In this chapter, the thought process involved in the development of this bachelor's work will be overviewed. Also, the ideas that led to the development of this bachelor's work will be mentioned. Additionally, further detail will be given about the objectives that were initially considered, and those that were finally fulfilled, realistically.

1.2 Motivation

Nowadays, technology advances at a very fast pace, introducing new technologies and techniques to apply in almost every field imaginable. The Merriam Webster dictionary defines the concept of *caring* as "feeling or showing concern for or kindness to others". Related to this topic, we can find that assisting other people is many people's objective.

Aging is an issue that is increasing progressively in many countries. The global population of the world is expected to double in the next half century ([1]). The age range that is expected to mostly grow is the population over the age of 65 ([2]). This population included total of 357 million people in 1997, and it is estimated that a total of 761 people over 65 will be alive by the year 2025 ([2]). Also, aging will have economic impact on business and will also affect the economy of nations. Longevity will impact the amount of money destined to the research and are of diseases and illnesses caused mainly by aging. For example, Alzheimer's disease ([2]).

Therefore, due to the human condition of aging, the senses are deteriorated over time. Falls are the second main cause of death or unintentional injuries in the world. An estimate says that there is a total of 424.000 people that die from fall-related injuries ([3]), such as being unable to get up from a fall, breaking parts of the body that disables these people from requesting immediate assistance. There is also a high percentage of these people that is over 65 years old ([4]).

In the medical field, there are currently many systems for monitoring patients when in need. For example, there are systems that allow monitoring vital functions of the body of a patient. Examples of this include medical alert systems that are designed for domestic use ([5]; [6]; [7]) which use a button as an activation mechanism for triggering notifications to emergency services. Other examples include medical systems such as systems that attempt to detect falls from patients. Often, these systems use proprietary software, developed by the company that distributes

the medical alert system for the detection of these falls. These fall detection mechanisms, such as [8], allow helping patients in a more automatized way, without the user interaction needing to press a button. Both these systems are leveraged in Section 2.1.1 and Section 2.1.2, analyzing their advantages and disadvantages in more detail.

However, one thing all these monitoring tools and techniques have, is that they are often expensive, as either the hardware or the technology used for this monitoring is not cheap, or they need additional human interaction for completing the required assistance. Therefore, an innovative, cost-efficient and intelligent software system seems to be a logical step towards obtaining a better, automatic way of taking care of patients' health conditions.
This bachelor's work is divided into the following process:

1. The development of an *offline* statistical study, in order to determine the different ranges of heart rate's variability in patients. For this, one own-made Android application and three Xiaomi Mi Band 2 devices are used, to measure the heart rate of test users. The frequency of measurements is once per minute, for a period of three days. After the study is done, all data and obtained information from the Xiaomi devices is correlated with the information provided by the users through the use of the Android application.
   The statistical study is three days long and initially involved a total of 36 users. In the end, a total of 24 users had introduced data correctly, and this data was able to be extracted from their mobile devices. There were some technical issues during the data extraction process (Section 3.7) and handling of data that reduced the initial number of users from 36 to 24. These users had their data dismissed for several reasons which impeded the use of their data (see Section 3.6).
   The purpose to do this study was to be able to tell the difference between a person doing physical exercise and, for example, sleeping. Issues such as trying to predict whether a user is having a panic attack were also an initial idea. However, as these events can't be predicted nor studied, an approximation has to be done based solely on the user's predicted heart rate when similar activities (such as physical exercise) happen ([9]; [10]).
   Finally, all obtained data will be used to implement a Machine Learning mechanism that will be able to detect which type of activity a user is performing at any given time of the day, and improve the interaction between the social robot and the patient by introducing interactions related to this topic.

2. Once the information is extracted from the statistical study, an algorithm that considers the age and gender of any patient is developed. This algorithm takes into account all the conclusions obtained from the previously developed statistical study. It will attempt to detect dangerous heart rate levels of the patient, and raise an alarm in case this danger level remains constant. This algorithm would be able to recognize and differentiate the aforementioned situations in point 1.

3. Finally, the algorithm and predictor are integrated into a social robot. Thanks to the *modular* interaction with the robot, the patient will be able to communicate whether an emergency is happening or not. In case of an emergency happening, the social robot will use software techniques, such as interaction with Application Programming Interfaces (for example, creating a Telegram

bot that responds to requests to obtain the user's heart rate from anywhere in the world in real time (5.4.1)) to alert doctors, family members, or even emergency services. A detailed description of these software modules can be found in Section 5.4.

1.3 Objectives

The objectives of this bachelor's work are:

- Implement a social robot able to transparently communicate with a user that is in distress and needs immediate help, as there are no current implementations of a social robot with this purpose nowadays. This will be explained in much more detailed in Chapter 2.
- Develop a static algorithm to detect subtle differences in a patient's heart rate, to determine if a patient is in danger or needs assistance
- Through the use of Machine Learning algorithms, generate a classification model that can conclude, in real time, the predicted activity a user is performing at any given time, based on the user's past and current heart rate measurements.
- Being able to help others, in a cheap and effective way, with a tool that can be worn by any patient to log heart rate information.

1.4 Regulatory Framework

In this section, we will mention the set of technical norms and restrictions that occur upon the published software. The published software regards both the Android application, developed and explained in Chapter 3 for collecting and cross-referencing data from the user, as well as the final robot application, explained in Chapter 5.

1.4.1 Legislative Analysis

The tool used for the development of the Android application is Android Studio. Therefore, it is subject to the license and agreement of Google's SDK. According to their terms and conditions, the software application is subject to the terms of the license agreement. Google grants the application developer a limited, non-exclusive and worldwide license to use the SDK for the purpose of development of Android-compatible applications.

With regards to the robot application explained in Chapter 5, it is developed under the framework of Robot Operating System (ROS). The core of ROS is subject to a three-clause BSD license. Therefore, the development of the software application is subject to the following clauses ([11]):

1. "Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer".
2. "Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution".

3. "Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission".

The development of the rest of the robotic software has been developed under the license used by the Universidad Carlos III de Madrid (UC3M), therefore it is subject to the following license:

1.4.2 Technical Standards

With respect to technical standards used for the development of this bachelor's work, the only standard followed for writing code in Python was **PEP 8**.

PEP 8

PEP 8 is a coding standard or document that specifies code style for Python. It includes differences from other languages, such as:

- Surrounding top-level functions and classes with two blank lines.
- Using underscores instead of capital letters for naming conventions.
- Surrounding method definitions inside classes with a single blank line.
- Establishing an indentation standard for Python, as Python is an interpreted language that relies on indentation.

Chapter 2

State of the Art

In this chapter, previous and current technologies related to the medical care will be discussed. First of all, the current situation of medical alert systems will be mentioned, focusing especially on domestic medical alert systems and, more specifically, button-pressing activation systems. Finally, all current used technologies, in software, firmware and hardware used, in regards to wearable devices, will be explored.

2.1 Current Situation

In this section, the current situation of medical alert systems is explored. Since medical alert systems are often used in domestic or personal households, as well as hospitals, these will be the studied domains.

2.1.1 Domestic Medical Alert Systems

Modern day technologies in hospitals usually allow monitoring of vital information of patients in an analogical way - without the use of technology -, as well as in a wired, technological way. This has some benefits - for example, the reliability that comes from using cables and wires to interconnect hospital components - but many more inconveniences - for example, the impossibility of monitoring vital health information, such as heart rate, blood pressure and/or patients' emotional state in a distributed way. This allows for a better organization and management of information.

In many countries such as Spain, elders - people of old age - and those who suffer from medical conditions, are often supplied with control systems that allow them to notify authorities when there are situations of distress in their lives ([12]). These conditions can either be physical (heart problems, people who have suffered myocardial infarction) or mental (early stages of dementia, Alzheimer's disease). However, these devices are often analogical or hardware-based and don't include any logic or software in their functioning, making them less useful when it comes to aiding people. This implies some disadvantages, such as:

1. Some of these devices are required to be carried by the patient, ideally at all times, in a specific manner that may not be comfortable for the patient, e.g. carrying it as a necklace, and may lead to problems such as forgetting it when changing clothes, or when going to bed. This problem is one of the most significant ones, as this design decision can negatively influence in the detection of vital information of a patient by decreasing the amount of time the patient carries the detector.

2. If the device's activation system is based on a button, which is often the case ([5]; [6]; [7]), it causes accidental presses and false positives. Depending on

the amount of force that the button needs to be applied, this can be regulated, however it entails some other design issues. If the button is too hard to press, or too small, people of old age may have difficulty, caused by diseases such as rheumatoid arthritis, and/or rheumatism. On the other hand, if the button is too soft, the amount of false positives caused by the accidental pressing of the button increases. Even if a middle-ground is attempted to be achieved by the designers of these systems, the system's detection accuracy can be compromised. Note that most current medical alert systems fall under this category of being triggered by the press of a button.

3. Some of the devices are not waterproof. Due to this design decision, the use of these devices and detection systems is compromised under some circumstances, such as daily hygiene routines of the patient, where water is involved and may cause disruption to the physical state of the device. For instance, showering or, depending on the position of the activation system of device, washing the patient's hands or feet. In the household, it is expected that patients carrying the device take a shower or bath routinely. Therefore, having a preliminary non-waterproofed device is something to consider when deciding the physical design of the detection system and hardware. The main reason why this would be an inconvenience in the design is the fact that patients would need to temporarily remove the device when performing daily hygiene tasks, causing the patient to become unprotected of the benefits of the detection system during the duration of these tasks. In case of a fall, this could lead to the inability to request help to medical services or automatically detect the need of medical assistance readily. A more detailed description of the waterproofing specifications will be shown in Section 2.2.1.

These aforementioned analogical technologies are often inexpensive, which can be considered as an advantage of having such a system. However, when it comes to patient's health and well-being of everyone in need, it is sometimes needed to add a marginal increase in cost to improve the system and its reliability. This is one of the main motivations of this bachelor's work: to develop an inexpensive control system for vital functions of a patient, with the purpose of detecting stress caused, for example, by falls or stressful situations that increase the heart rate of a patient over a small interval of time. Additionally, human-robot interaction will be made with a social robot, to communicate with the patient in real time, obtain responses, and act upon them.

2.1.2 Fall-Detection Medical Alert Systems

Current personal household systems usually include fall detection systems in their preexisting alarm system as an additional feature that can be agreed upon. These systems usually use proprietary software and fall detection mechanisms based on more general methodologies ([13]; [8]).

These systems have several advantages:

1. They are connected to a control center where assistance personnel have the possibility to communicate directly with emergency personnel such as fire and police departments.

2. Sensors will work with high accuracy based on proprietary-based detection mechanisms, if the patient is within the range of the sensors.

3. The systems include additional support for buttons, connecting the medical system to the landline.

4. There are always people available for aid.

And disadvantages:

1. They are unable to detect a fall whenever a patient is not in the range of detection of the sensor of the alarm system. A solution for fixing this disadvantage would be to equip the house with a greater amount of sensors surrounding the household completely. This can incur in a significant increase of pricing when installing the system.

2. In case the patient decides to include additional support for buttons, and some of them are wearable, the patient may be subject to all the disadvantages previously discussed in Section 2.1.2 about button-pressing activation devices. Also, wireless buttons may suffer from battery inconveniences, such as having to recharge the devices periodically.

3. The systems are much more expensive than simpler alert systems.

Button-Pressing Activation Systems

More specifically, inside the personal household systems, many designers and businesses attempt to make the use of the device as clear as possible to the patient and their families, by just implementing a button that may be considered dull by the patient because of its simple design. This design idea has happened since early designs and productions of hardware devices for this purpose ([14]; [15]) and has been refined over the years to allow additional functionality ([7]). Initial ideas included pager systems with additional panic buttons that could be activated. However, with technological advances, designed systems allowed wireless communication with a remote central station. Finally, more recent designs expand this communication with more recent paradigms, such as WiFi or 2G/2.5G/3G/4G internet communication ([16]). The implementation of a panic button is also considered, as a location tracking mechanism ([16]), however there is no further real-time analysis that can aid a person in real-time, in the moment of an emergency.
Devices, as often happens, can include an alert button which the patient can activate or press, in order to inform an emergency system of a harmful or disruptive situation or medical condition. Also, in order to prevent false-positives, an important design issue that needs to be considered, the device can also implement preventative measures that require the patient to press the button in a specific order or sequence, for example, the pressing of the button successively 3 times, or alternatively include a preventative measure which consists of confirming whether the button press was intentional or not. This would be able to be implemented in devices with a greater range of patient-input buttons, such as a menu in which the patient can select, go right, left, up and down with arrows or other designated buttons. As we have previously mentioned in Section 2.1.1, some advantages and disadvantages of this are the following: Advantages:

1. Ease of use due to the simplicity of the design of the wearable device.

2. Low cost of hardware.

Disadvantages:

1. Accidental false positives where the patient is doing a task and pushes the button against a door, a wall... This can also happen while in the higher-consciousness levels of sleep.

2. Possibly, intentional false positives, caused at will by the patient because he/she is disoriented or does not remember the purpose of the button. This can happen to patients on early or advanced stages of dementia.

3. The device can be considered a burden by the patient, which discourages him/her to continue wearing it.

4. Immediate medical assistance can not be provided with some of these systems, as only location tracking is implemented on the event of the pressing of the panic button ([16]). Also, the current state of the patient can not be measured in real-time, medical practitioners are only warned that a button has been pressed, with no additional information of any kind.

All the advantages and disadvantages discussed in this section need to be considered when doing the physical design of a new device. It is also observable that, the scientific proposals for real-time aid of a patient are not clear, and they include non-vital information of the patient, such as its location, without looking at information that may be useful in an emergency situation, such as a history of the patient's heart rate from the moment of an alert to medical assistance arriving,

Button-pressing activation systems that are specifically encapsulated under the intended use in personal households, establish some criteria in order to select people in need of these devices. In case the economic capabilities of the patient are not enough to finance the device themselves, the following criteria is considered ([12]):

- It is considered whether the patient lives alone or spends most of the day alone without contact with other people - physically.

- The mental health of the patient is also considered, making specific emphasis on whether the patient has anxiety or anguish caused by its loneliness and isolation from society

- Risk of a medical emergency based on the age, possible disabilities of the patient and/or a current or chronic illness.

- Low economic resources of patients, and/or of the family of the patient.

There are several things to consider on these systems in regards to their cost and its efficiency when it comes to results. During the whole existence of personal alert systems that are connected to emergency systems, there does not exist enough evidence on the robustness and the economic viability of these systems for the manufacturers and maintainers of the system, and in Europe, there has been a raise in the amount of studies that are developed in order to identify critical evidence that supports its efficiency / inefficiency. The expansion of remote assistance in countries such as Spain has been intense during the last 20 years, and has been favored by the newly created policies to aid elders in need ([17]).
Additionally, in some cases, governments such as the one in Spain finances one hundred percent of the costs entailed in the maintenance and telecommunications assistance service of these devices for people over the age of 80, and for some cases, even below this age in some certain conditions. Also, the system can be freely provided for people over the age of 65 if their state of independence can not be proven - meaning, that they depend on other people for some parts of their daily habits.

Finally, there is a rising trend in the amount of responsibility delegated to the telecommunications centers that handle and act as mediators between a patient and emergency systems, being these, in some cases, the ones that arrive to the patient's household in case of an emergency to check on them instead of the emergency systems. This causes initially untrained people to be the first responders in emergency situations, which can be considered as a disadvantage of these systems ([17]).

2.2 Hardware, Firmware and Software

In this section, all hardware, firmware and software technologies used in the development of the bachelor's work will be explained. As these technologies all come together to form a system, but work independently from each other, a detailed overview of the system, part by part, must be given. In the following chapters (Chapter 3, 4, 5), these systems will be joined together and an explanation of how they work as a whole will be given.

Note that, from now on, instead of referring the Xiaomi band devices with their hardware IDs, we will use the following pseudonyms:

- ED:61:38:85:D1:FC, which will be called *Device 1* from now on.
- D8:59:59:8A:E5:69, which will be called *Device 2* from now on.
- E6:A1:BD:94:31:54, which will be called *Device 3* from now on.

2.2.1 Waterproof Hardware

Hardware components sometimes require the capability of being waterproof. When talking about a device that mostly relies on hardware to properly function, and may be submerged under water in certain circumstances, the waterproof capability is crucial. Patients who will be wearing this device may be exposed to water or humid conditions during the use of this device.
As one hardware component of the bachelor's work is a set of Xiaomi Mi Band 2 devices, a consideration into the hardware capabilities of this device must be given. Waterproofing systems used for most of the wearable devices present in medical and non-medical devices are certified on either IP-67 and/or IP-68 [18]. The IP was created by the International Electro-Technical Commission (IEC), and it is used to determine the degree of resistance of a device based on fresh water and/or particles. Other fluids or liquids are not contemplated by this standard (for example, soda, beer, acids... It doesn't contemplate either the addition of components into fresh water such as salt, which makes up salt water). It also contemplates the resistance of devices to other raw materials such as dirt and dust.

The first digit in an IP-XX certification means that, for a period of eight hours at a time, it has been tested and proven that the device is resistant to a specific kind of material. The following table represents the first digit meanings of an IP-Certified system ([19]; [20]):

The second digit represents the degree of water resistance, which is particularly interesting. Nowadays, for this second digit, both the 7 and 8 are leading ratings in the industry. Number 7 represents that the system can be submerged in up to one meter of water for half an hour, and number 8 represents the ability of the device to be submerged under 1.5 meters for half an hour. The other codes (1 through 6) also have meaning but are irrelevant as they are no longer contemplated in any

IP Code	Protection
1	Protection from contact with any large surface of the body.
2	Protection from fingers or similar objects.
3	Protection from tools, thick wires or similar objects.
4	Protection from most wires, screws or similar objects.
5	Partial protection from contact with harmful dust.
6	Protection from contact from harmful dust.

TABLE 2.1: IP Codes for the respective amounts of harm caused by materials to a IP-certified system (1st digit).

IP Code	Protection
1	Protection against vertically dripping water
2	Protection against vertically dripping water with a <15º degree tilt
3	Protection against vertically dripping water with a <60º degree tilt
4	Protection from sprays and water splashes in all directions
5	Protection from low-pressure water projected from a nozzle of 6.3mm of diameter
6	Protection from low-pressure water projected from a nozzle of 12.5mm of diameter
7	Protection from water immersion with a depth up to 1 meter or 30 minutes
8	Protection from water immersion with a depth of > 1 meter

TABLE 2.2: IP Codes for the water resistance of an IP-certified system (2nd digit)

water-resistant wearable device: not on Xiaomi Mi Band 2 or its competitors. The protection standards against water usually require a rating of either 7 or 8 in devices, because the lower ratings have worse qualities that the consumer takes as granted - for example, the protection against vertically dripping water (IP code 1), or protection from sprays and splashing of water in all directions (IP code 4) - ([19]; [20]).

A combination of both explained numbers construct the IP-67 and IP-68 standards, whose differences are how deep the device can be submerged underwater (less or more than 1 meter).

2.2.2 Bluetooth Technology

Bluetooth Technology is used in several wearable devices as well as smartphones and personal computers as a solution for wireless device communication, using OTA (Over the Air) transmission ([21]). This technology is based on an ad-hoc network, meaning, a decentralized type of network without a structure, that builds this structure over time with the devices that interconnect with each other. The use of this technology in the communication of smartphones and wearable devices is common practice nowadays. This is due to the hardware required to communicate devices being ideal for devices that can't be wired, because of the nature of the wearable device and how it's used; and for an unstructured type of network. Therefore, no third party certification authority is required for connecting to another Bluetooth device.

The Bluetooth technology is used in Xiaomi Mi Band 2 devices, as a means to communicate all the information. More specifically, all used Xiaomi devices use a technology called BLE (Bluetooth Low Energy) is used. Depending on the manufacturing region of the Xiaomi Mi Band 2 device, two different Bluetooth protocols are included. For the international version, Bluetooth BLE 4.2 protocol is implemented. However, for China-manufactured devices, the standard 4.0 Protocol is

implemented. BLE technology is commonly used in other fitness trackers, due to its advantage with the standard Bluetooth 4.0 protocol, whose power consumption is greater and not suited for devices such as a wearable device; the problem increases when the sending of information to the mobile phone and its frequency is increased (which, in case of the final, real-time application, occurs). Therefore, choosing Xiaomi Mi Band 2 was integral for this study, and purposefully chosen for having BLE instead of the standard Bluetooth 4.0 protocol integrated into the wearable device's system, amongst the already-mentioned decisions in Section 2.1.1. Another difference, although less important, is that BLE allows for a connection bandwidth of up to 250Mbps, instead of the standard 25Mbps in the case of Bluetooth 4.0. However, since the information I use in my bachelor's work is limited and doesn't nearly compare with these ranges of bandwidth, this positive quality was less important.

Finally, the Xiaomi Mi Band 2's Bluetooth counts with a Bluetooth that is categorized as Class-2 Bluetooth ([22]; [23]). This refers especially to the operative range of the signal, and is directly related to the transmitting power of the Bluetooth transmitter.

The main differences between the three existing types of classes are:

Device Class	Transmission Power	Intended Range
Class 3	1 mW	Less than 10 meters
Class 2	2.5mW	10 meters <=> 33 feet
Class 1	100 mW	100 meters <=> 328 feet

TABLE 2.3: Bluetooth device transmitter classes

In the case of the Xiaomi Mi Band 2, we have a transmission power of 2.5mW, which equals about a range of 10 meters. This was also thoroughly contemplated when deciding which wearable device to use. As the distance between a social robot and the patient with the wearable device needs to be enough for a typical-sized household, the transmission power of other devices were shown. Additionally, this transmission also needs to avoid causing issues to the transmission of data between the device and the monitor / social robot.

2.2.3 Display Technology

In this section, the display technology of the hardware will be discussed. One of the focuses of choosing an ergonomic and easy-to-use device is the ability for patients to use the wearable device as a standard watch. Therefore, a display technology that allows users to clearly see the screen n is important. In the past, many devices used the standard LED (Light-Emitting Diode) technology. It has been, and still probably is, the most common type of display available in the display market, including televisions and smartphones. There is a common misconception between the use of the LED display and the LCD display. Most smartphones and televisions use the LCD (Liquid Crystal Display) technology as well, whilst the LED technology only refers to the lighting source of the screen, and how this affects its performance.

In the current market, there is a technology called OLED (Organic Light-Emitting Diode), which is nowadays used in higher-end smartphones, and wearable devices are also catching up. Examples of the use in higher-end smartphones are the most recent iPhone X and XS versions. Other Android equivalents of higher-end devices are the Google Pixel 3; and, in the TV world, we can also see the LG C8 display that features this technology.

Both LED and OLED technologies have differences between each other:

1. Pixel illumination: the main concept to differentiate them is that LED displays emit light to illuminate pixels on a screen, while the OLED technology pixels produce their own light source.

2. Brightness: With respect to brightness, LED LCD displays are brighter than OLED displays, which is something to consider on smartphones and televisions. This also causes a higher battery use (when we talk about smartphones and wearable devices) or a higher power use (televisions). This has also been a determining factor when choosing which kind of device was better to represent and be chosen to implement part of the system of my bachelor's work.

3. Color: directly related to brightness, the color, as it is in all displays, is a 3-component RGB element for each pixel in the display. Since the brightness is lower for OLED displays, colors appeared less realistic on early-age OLED televisions; nowadays, this problem no longer appears. These days, the common problem with color is its *volume*: the saturation of the color appears to be less realistic on OLED displays than LED LCD displays. However, for the purpose of this bachelor's work, this isn't an issue since the only color being displayed is white, and color volume is not an problem. With respect to the directly-related brightness, upon testing the Xiaomi Mi Band 2, there doesn't seem to be any issues when reading the OLED display, unless the device directly staring down the sun.

Many modern wearable devices nowadays use an OLED technology. Comparing to its predecessor Xiaomi Mi Band 1, which used a standard LED display, Xiaomi Mi Band 2 uses OLED technology for its display. It's also been studied that LED and OLED have differences with respect to performance ([24]).

Thus, this display technology comparison was useful for choosing the Xiaomi device, taking into account battery consumption and brightness, and how this will be beneficial for patients.

2.2.4 Sensor Technology

Sensor technology is something very necessary in the domain of a wearable device. We usually can distinguish between three types of sensors: cameras, accelerometers and vibration actuators.

In many modern wearable devices, these are the main components that are needed in order to obtain external information from the real world. Also, their small sizes make it perfect for an ergonomic design that doesn't exceed in size and makes the design cumbersome. Devices such as the Xiaomi Mi Band 1, 2 and 3 all have cameras, accelerometers and vibration engines. These so-called *cameras* are not cameras in the traditional sense of the word. They obtain information from the outside world with a technology called *photoplethysmography*.

In the Xiaomi Mi Band 2, we have a **photoelectric heart rate sensor**, which also receives the name of a PPG (photoplethysmography) sensor, which measures the heart rate of the user by shining light into the skin of the user, and calculates the amount of light that is reflected and scattered by the blood flow. This is based on the principle that the light that comes into the body, more specifically located in the

wrist area, will scatter depending on the blood flow changes predictably. The PPG is composed of several components, at a lower abstraction level:

- Optical emitter, which sends lights, typically green - such as what happens with the Xiaomi Mi Band 2 - in order for the optical emitter to work properly with all kinds of skin tones. Xiaomi Mi Band 2's optical emitter is composed of 2 LEDs.
- Digital Signal Processor (DSP) which is in charge of analyzing the scattered light and translating this information into human readable heart rate values that we can make sense of.

The purpose of accelerometers and photoelectric sensors are especially contemplated in this bachelor's work. Accelerometers measure the acceleration of the device at any given point in time with respect to the device's instantaneous rest frame. They serve as measurers for the indirect amount of activity a user is doing. This can vary between users that have a different stride, or a different way of walking. For example, a patient with a broader arm movement will give different accelerometer values than a patient with a narrower arm movement. Also, this indirect amount of activity will vary depending on the type of activities the user does on a daily basis, i.e. a construction worker, who repeatedly uses his/her arms for many of the things at work, will have different activity records than a user with a sedentary line of work.
Photoelectric sensors are, on the other hand, typically used to gather heart rate information in wearable devices. Because having small photoelectric sensors does also imply that the quality of the image recorded by the sensor can't be as good as a higher-quality photoelectric sensor, the use of these photoelectric sensors in wearable devices is typically destined for this kind of activity.

2.2.5 Firmware Technology

In this section, the firmware specifications of the Xiaomi device are described. As firmware is essential in embedded systems, and all wearable devices fall under this category, firmware technology must be detailed as it is an important part of the functioning of the system.

The firmware, which can be defined as precompiled software embedded into the hardware of the system, is integral and essential for the proper functioning of any wearable device. Since the firmware, once distributed, shouldn't contain any error, and updating the firmware in already-distributed devices is complicated and may not be installed into every device, as it is the discretion and responsibility of the consumer to do so, firmware updates are rare.

In all wearable devices, the firmware is normally changed when a new batch of products is manufactured, and updates are seldom. In Xiaomi Mi Band 2, there are several firmware versions, each of which has its own updates. However, having a version of the firmware also requires to have its equivalent proprietary-based software of Xiaomi for wearable devices, called Mi Fit. Without this software, updating the firmware is very hard and downgrading it is impossible. Here is an example of Xiaomi Mi Band 2 firmware changes between versions, seeing how some new functionalities are implemented and how the Mi Fit version must also change in order to properly connect the device to a smartphone:

Version	1.0.1.54	1.0.1.47	1.0.19
Activity Recognition	No deep sleep	No deep sleep	Works
HR Measurement	Works	Works	(No HR Measurement)
Mi Fit Version	3.1.0	>2.4.0	(No app)

TABLE 2.4: Xiaomi Mi Band 2's firmware changes between versions ([25]).

2.2.6 Software Technology

When talking about the software on most wearable devices nowadays, it varies depending on the manufacturer, even though the functionalities implemented by the software are very similar. The software can be decomposed into two parts: the one that allows communication with a smartphone, and the one that implements the functionalities in order for the wearable device to acquire some kind of intelligence.

With respect to the functionalities, all code is proprietary by the manufacturers, and without further inspection, nothing can be analyzed. In the case of the Xiaomi Mi Band 2, we can say that the only recognizable part of the software is the storage system, local to the wearable device until it is downloaded into a smartphone using the proprietary Xiaomi application or other third party applications that allow this as well. Information in natural language can be accessed through the use of official - or unofficial - applications that automatically process the data in the database of the wearable device. The set of information that can be observed in the database - a PL/SQL structured database - will be further analyzed in Chapter 3, where an analysis of the database structure will be made. Also, a data storage efficiency analysis will be performed, analyzing the amount of data that the device stores, and seeing how all this impacts at the performance.

2.3 Tools

In this section, the set of tools and libraries/modules that have been used for the development of the code for the social robot will be explained.

2.3.1 Programming Languages

For the development of the code, two standard programming languages were chosen: C++ and Python. C++ is a general-purpose compiled programming language that efficiently allows programmers to incorporate programming solutions to problems. It is statically typed, and was developed by Bjarne Stroustrup as an extension of the original C programming language ([26]). C++ is often considered the equivalent of C with classes, as well as other additional improved functionalities, such as an improvement in the standard library, that allows using data structures very simply, as well as strings instead of arrays of characters in C. The C++ programming language will be used in this bachelor's work partly in the social robot's code. Therefore, the C++ code will implement, in this bachelor's work, a producer, in a producer-consumer architecture. Python, which is an interpreted high-level programming language, was released in 1991, and was originally designed by Guido van Rossum. It provides many simplicities not found in other programming languages and the use of modules / libraries is typically more convenient than in other

programming languages. For this, and because ROS (Robot Operating System) supports both C++ and Python, both languages are used in the development of the social robot's code, in order to learn and showcase my knowledge in both programming languages simultaneously. Therefore, the Python code will implement, in this bachelor's work, a consumer, in a producer-consumer architecture. More details about the used architecture will be discussed in Chapter 5.

2.3.2 ROS

In the domain of the software used indirectly on robots, there are several industry-level Operating Systems used for the production of robots. Typically, it is common practice that companies use their own built-in Operating System for the development of industry-level robots, because hardware and software needs to pass certain certification requirements. Controlling both the hardware and software parts of the robot is the only way to predict the operations of a robot at all times, and ensure that no critical errors occur. This is especially important when talking about robots that operate in the field of Real-Time Systems, more specifically critical real-time systems, where a missed deadline or the improper functioning of a part of the system can cause harm to humans, or in some cases, even death. However, there is one well-known open-source alternative to all the proprietary OS's used by the companies: ROS or Robot Operating System. This OS is a robotics middleware that provides the system with hardware abstraction that can allow companies to develop the software part by abstracting the hardware partially or fully. It is not technically an Operating System, but it includes many features that allows inter-process communication; as well as package management and hardware abstraction amongst its many features ([27]). It allows a very homogeneous and simple communication between nodes over the Internet or in a distributed manner.
In the case of this bachelor's work, ROS has been chosen as the standard Operating System on top of the robot. This was not arbitrarily chosen, as the licensing of this OS is not necessary as it is open-source, and it is very convenient for the development of social robots. The department of robotics and automation systems of the Universidad Carlos III de Madrid, RoboticsLab, have developed all robots to have ROS as its underlying middleware, running between Linux and the robot's hardware.

The development of robots can rely on ROS as its middleware, because of its many advantages. Some applications may need to implement:

- Image processing or analysis, including video or other forms of interaction with sensor cameras, which can be aided by the use of the OpenCV library in both C++ and Python, for which ROS is a possible candidate. OpenCV stands for Open Computer Vision, and is a library developed specifically for image processing and rendering.
- AI, machine learning, deep learning and other related modules, which can be found in the Python programming language is also prepared for, and ROS runs on top of it;
- Any other possible, imaginable, software requirement that can be satisfied with either C++ or Python, is valid and implementable underneath ROS.

Note that ROS is a robotics middleware and therefore is not capable of supporting operating system basic concepts, such as scheduling, memory, storage, file system and processor management... It must run on top of an Operating System. In case of ROS, it is supported to run on **Ubuntu** distributions; it is therefore limited to run and

be updated on a 5-year span (which is Ubuntu's Operating System updating policy) for security reasons. However, from Ubuntu 18 onwards, this policy will change to a 10-year lifespan. Therefore, for the development of software using ROS, the choice of distribution is important. In case of this bachelor's work, all software has been implemented using both Ubuntu 14.04 LTS and **Ubuntu 16.04 LTS**, being the latter the one in which all experiments and tests are made, for the aforementioned security and lifespan reasons. The architecture chosen is **64-bits**.

However, some modules can only run on 32-bit machines (such as the Nuance speech recognition software currently present as an option in the speech-recognition system of the social robot). Therefore, these special 32-bit modules will run inside a 64-bit Docker container, which will communicate as a 64-bit ROS node to all other ROS nodes in the system of the social robot.

Chapter 3

Data Collection Application

3.1 Purpose

The purpose of having a data collection application is the collection of data from test users. This data is captured in real-time using a Xiaomi Mi Band 2 device. The data does not require an Internet connection, as the process of recovering all the stored data is done via Bluetooth through a smartphone. Users are required, after the development of the Android application has been finished, to use the application for the purpose of logging their daily activities and habits through the use of a SQL Database. This application is installed in Android device of the user, in a complementary way, together with carrying a Xiaomi Mi Band 2. The wearable device will monitor the user's heart rate simultaneously over a period of three days. More information about the Xiaomi Mi Band 2 study will be discussed in Chapter 4. After the data collection process has been done, the data will be extracted from the mobile device through the use of Android Debug Bridge (ADB) related tools. More specifically, the tool that will be used is Android Studio. It subsequently requires the Android Source Development Kit (SDK) to be installed, which is a set of tools that allows to extract and display information from several devices. It will be used specifically for extracting information such as the database where all data will be stored, via USB connection, through Android Studio's Graphical User Interface (GUI). Afterwards, this information will be studied. The kinds of information that are and can be extracted from the user's database are:

- The user's type of activity. There are several activities which the user can indicate: Rest (Reposo), Walking (Caminar), Physical Exercise (Ejercicio Físico), Sleep/Rest (Dormir) and Others (Otros). The 'Others' activity is created for indicating all other activities the user is doing but is unsure whether the activity may fit. For example, showering or working in an environment where the user stands up and sits down very frequently.

- The time in which this activity has started. The user was explicitly asked to indicate the activities when they were started - preferrably - because this could benefit in the future study of data to measure the heart rate changes at this time.

- The duration - or an estimate - of the activity. Users were asked to try to be as accurate as possible and do their best when logging information. It is understandable that it can be cumbersome to log all information with high frequency, but this was very important for the study. Users were asked mainly to indicate major changes in their activity - for example, sitting at home and preparing to go to the gym; walking in the street and sitting down -. In the analysis phase,

we will consider that users may not have been completely accurate and precise on these annotations about their activities.

There are two different parts of this study: the data collection and analysis processes:

- First, the data collection process, that uses an Android application to log user activities while they are wearing a wearable device. This part could also be caled the *Android application part*, and is performed in this chapter. The Android application created for this purpose is named the Medical Logger application.
- Secondly, the data analysis process, that focuses on the data from the Xiaomi Mi Band 2 (accelerometer and heart rate values with a timestamp). After extracting this information, it is analyzed. This part can be also called the Xiaomi Mi Band 2 part, and is performed in Chapter 4.

Therefore, all the information previously mentioned in 3.1 is stored and kept until both parts of the study have been completed (the Android application part, and the Xiaomi Mi Band 2 part). Then, a more comprehensive analysis of activities, timestamps and heart rate frequencies during these activities will be carried out in Chapter 4, by correlating and making conclusions from both parts of the study.

3.2 Android Application Requirements

In this section, the requirements of the Android application are described. For the purpose of having a easy-to-use, simple application, we gather the following requirements:

1. The application must be *error tolerant*: when a user indicates a duration of an activity that is not the one they desired to input, they would have the opportunity to overwrite the last introduced value by clicking another button, or through the interaction with a progress bar. Another example is if the user accidentally accessed a menu that was not desired. In this case, the user would have the option to return to the previous menu with built-in navigation buttons, apart from the ones already existing in the mobile device.
2. The application must be *resilient*: in the event of an exception launching, through exception handling code, the error would not cause the application to crash. For example, having an exception handler for changing the orientation of the mobile device from vertical to horizontal inside the application. This would normally break the design of the Android application and launch an exception; however a new layout is launched when this happens.
3. The application must be *easy to use and minimalistic*: the UI design is intentionally made very simple not to overwhelm or distract users from the original purpose of the application, which is introducing activity data. As a result, the application does not have any color that can be considered distracting, or any button that is not required for the specific functionality the application was designed.

3.3 Architecture

In this section, the architecture used for the development of the Android application is explained. This architecture is based on the concept of *activities*.

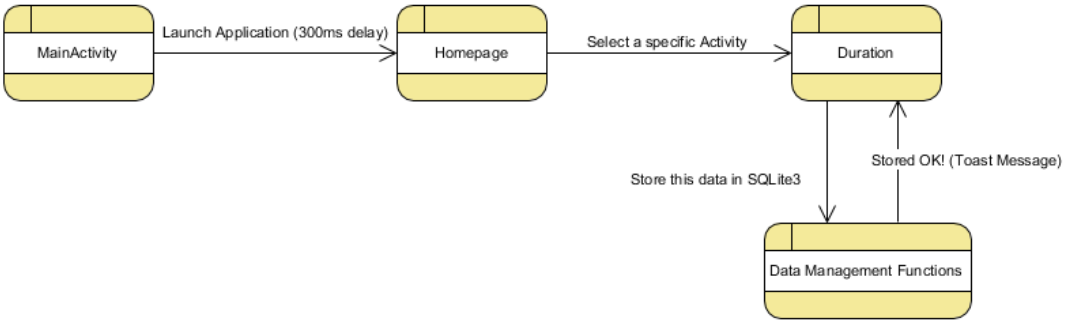


FIGURE 3.1: Android application Lifecycle ([28]). Here, we can observe an average execution of the Android application, where a user launches the application, chooses an activity and a duration, and confirms the selection.

3.3.1 Activities

Activities are classes that constitute the execution of an Android application at any given time. The activity lifecycle describes the different states in which an activity can be, depending on whether the application is in the foreground (in the GUI) or not. The Android system instantiates these activity classes by invoking specific methods, each of which represents a state of their lifecycle. The lifecycle of an activity can be shown in Figure 3.1:

These activities take control of the GUI temporarily, as long as the user is interacting with this activity. We can observe the static view of the system and dependencies between classes, through a class diagram in Figure 3.2.

The specific functioning of each activity can be found in 3.3.1. As we can observe in Figure 3.2, we have MainActivity, which has a 1-1 relationship with the Homepage activity. This happens because the homepage activity is launched directly after the MainActivity activity. There is not any composition or dependency involved either, as they do not share any data between them.
We can also see that we have a 1-1 relationship between the Homepage activity and the Duration activity. As we explained before, this is done intentionally this way, by only passing and exchanging data between these process with the name of the button pressed, in order to differentiate which activity the user is going to perform. More information about the set of data shared between these elements can be found in Section 3.3.3: Sequence Diagrams.
Also, we can see that the Data and DatabaseHelper classes has a 1-n relationship. This is due to the fact that the Data is an auxiliary element of DatabaseHelper and can be instantiated as many times as needed. In the implementation, one Data instance is created for each element to insert into the Database. That means that, if the user decides to press the confirmation button four times sequentially, four different Data instances will be created for its compilation and storage into the database.
Finally, we see that the Duration activity and the DatabaseHelper class is governed by a 1-1 relationship, as there is only one DatabaseHelper instance created, at the creation of the Duration activity, and remains that way until the Duration activity is destroyed.

We will shortly describe the functionality of each of the activities in the following subsections.

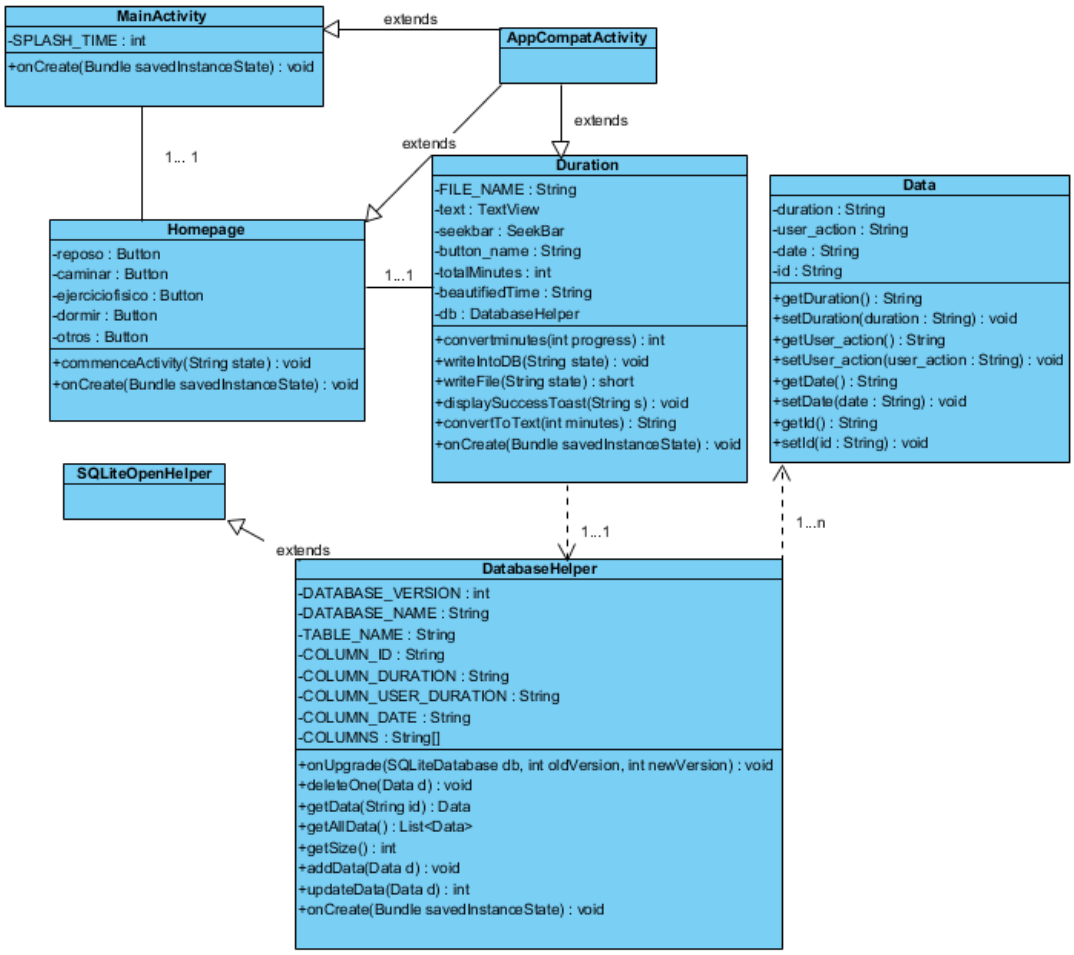


FIGURE 3.2: Class Diagram of Logger Android application.

MainActivity

This activity is the launcher of the application, meaning that it is always executed first when the Android application loads onto main memory. Its functions were very simple: it shows a splash screen for a duration of 300 milliseconds, with the icon of the application, and redirects the flow of the process to the next activity: the homepage.

Homepage

This activity was the main and most frequent activity as shown by users. Since all specific activity buttons were in this screen, users had to be in this screen in order to select their future/imminent activity to perform. If the user performed a click on a button, they would be redirected to the next activity (the *Duration* activity), with a additional parameter passed to this activity as text (the name of the type of physical activity performed by the user). This was implemented in order to avoid having to create five different activities for the duration, one for each type of activity. The execution of the Homepage activity therefore can lead to one activity, Duration, with five different String arguments. The type of physical activity performed by the user can be one of the following: *sleep*, *other*, *physical exercise*, *walking* and *resting*.

Duration

This activity is in charge of providing the user with a set of tools to determine, in an easy and intuitive way, the duration of the activity of the activity that they are going to perform. For that, there are two different possibilities:

- To indicate the duration with a progress bar, which implementation is made from scratch for indicating a range of time (between 0 minutes to a maximum of 10 hours). The progress bar advanced, and was designed to advance, one percent every 10 minutes. This was made for user interface (UI) simplicity, as it is not visually elegant to have non-multiples of 10 in the durations of the progress bar. *Note that the application was initially designed to include all possible amounts of time (3 minutes, 27...) however this design was replaced by the simpler option of having multiples of 10, because during the preliminary testing phase of the Android application, users reported this preference in the final progress bar.*
- To indicate the duration with four different radio buttons, whose durations were 1, 2, 4, and 8 hours respectively.

In case the user had indicated a time they did not desire; pressed two different radio buttons; or chose to first use the progress bar, but then wanted to introduce the final duration using the radio buttons, the latest introduced duration would be the one stored in the database when confirming the duration. So, the interaction between the progress bar and the radio buttons is made possible intentionally in order to have a simple user interface (UI) with *error tolerance*.

Once the duration was set, by any of the aforementioned ways, the user had to press a button to confirm the duration. In the event of a button press by the user, this information would be stored persistently in a SQL Database in the internal storage of the mobile device, using standard system calls to write into a file. The information logged into this file has already been mentioned in Section 3.1. When the user confirmed the button, a toast message appeared on the screen, to inform the user that the information was successfully stored. We can observe that in the third image in Figure 6.1. Finally, the user had the possibility to return to the Homepage activity, using a button that redirected the user to this activity, at the end of the activity's interface; or simply pressing the *back* physical button of the mobile device.

Data Management Functions

It is composed of two classes: the Data activity and DatabaseHelper activity. These classes do not have any view or interface associated to it. The reason why they do not have a GUI is because their purpose is to manage data structures and the connection with a SQLite3 database. Therefore, they are considered auxiliary classes that allowed others to perform more complex tasks. In this case, both these components act cooperatively to manage the SQLite3 Database that is operable by calling the functions of these activities. For this purpose, the Duration activity can instantiate the DatabaseHelper class and use its functions using an Object-Oriented Programming (OOP) paradigm, to commit the user input into the database. This is the action performed, on a lower abstraction layer, when users confirmed the duration of the activity, as mentioned in Section 3.3.1.

In the implementation of the Data class, the definition of the data that was stored in the database was defined. It included:

- An *ID* for the element stored, to avoid duplicate rows in the relational SQL Database.

MainActivity.java | Homepage.java | Duration.java | DatabaseHelper.java | AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="ronan.prairie.jasper.myapplication">
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="Logger"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".Homepage">
            <intent-filter>
                <action android:name="android.intent.action.HOMEPAGE" />
                <category android:name="android.intent.category.DEFAULT" />
            </intent-filter>
        </activity>
        <activity android:name=".Duration">
            <intent-filter>
                <action android:name="android.intent.action.DURATION"/>
                <category android:name="android.intent.category.DEFAULT"/>
            </intent-filter>
        </activity>
    </application>

</manifest>
```

FIGURE 3.3: Android Manifest of the Android application. In this picture we can observe all components explained in 3.3.2. All metadata, such as the name of the package of the Android project, as well as other data (name of the Android application, location of the application's icon) can be found here. Additionally, we can define different intents, with custom identifiers, to change context between activities.

- The *current date*, formatted into a human-readable format for simplicity.
- The *action* chosen by the user, which was previously passed as an argument from the Homepage activity to the Duration activity.
- The chosen *duration* of the activity by the user.

To summarize, both functions implement the necessary SQL statements to have a programming interface in Java-readable format and a database that persistently stored all the information introduced by the user.

3.3.2 Android Manifest

The Android manifest is a special XML (Extensible Markup Language) v1.0 file associated to every Android Studio project by standard. In this file, a lot of information can be found and specified about the set of activities. The Android manifest is crucial for the development of the Android application. We can see the Android manifest in Figure 3.3. It performs the following tasks:

1. Set additional names for the activities, called *intents*, which associated an activity with an Android identifier, of type *android.intent.action.XXXX*, where

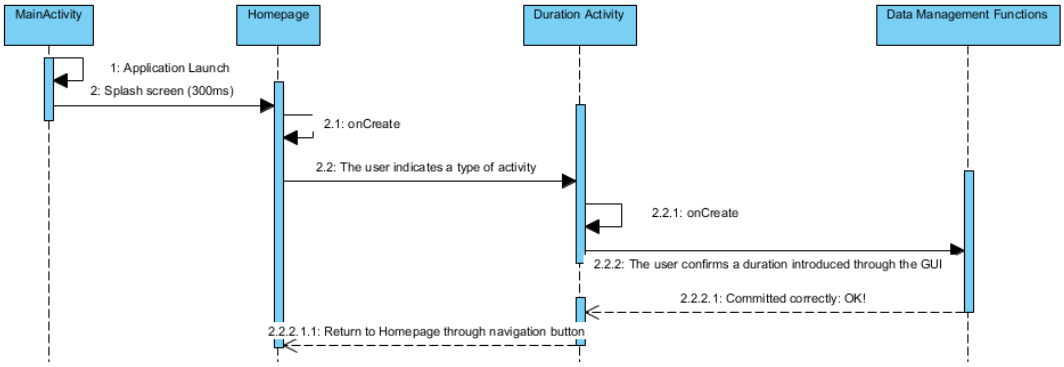


FIGURE 3.4: Sequence diagram of the execution of the Android application, activity Duration.

XXXX represented the name of the intent. This name could be anything, and would later let us call the startActivity() function in order to switch from one activity to another, simply using the preset intent name.

2. Indicate which of all activities shall be executed when the application launches, also known as the application launcher. This is defined modifying the category of the intent, of type *android.intent. category.LAUNCHER*, whereas all the other non-launcher activities need to be in a different group, of type *android.intent. category.XXXX*, where *XXXX* could be anything, as long as all the other activities are grouped inside.

3. Set the name of the Android application.

4. Set the default icon for the Android application.

3.3.3 Sequence Diagrams

In this section, the flow of all activities will be studied and graphically shown, to demonstrate how the usual execution of the application was handled. Using previous knowledge from software projects management, the following sequence diagrams were developed, and making use of UML (Unified Modeling Language) for having consistency in the designs. UML is a general-purpose language that allows the development and modelling of software systems. It provides a standard view of the design of the system, therefore it will be very useful here to comprehend the components and subcomponents of the Android application.

In this section, we will use a Data Flow diagram in order to see which kinds of information are shared in a *dynamic* way, emulating the real execution of the application. The execution will be method-specific. Aside from these general, high-level sequence diagram found in Figure 3.4, we can observe lower-level, function-specific sequence diagrams in Annex C. Herein, we can observe the set of high-level sequence diagrams, one corresponding for each method in the used classes:

3.4 Data Extraction

In this section, the process of extracting the database from the internal storage of the device of the user is explained, as well as the structure of the own-developed SQLite3 database to store user data. The process of extracting information had to

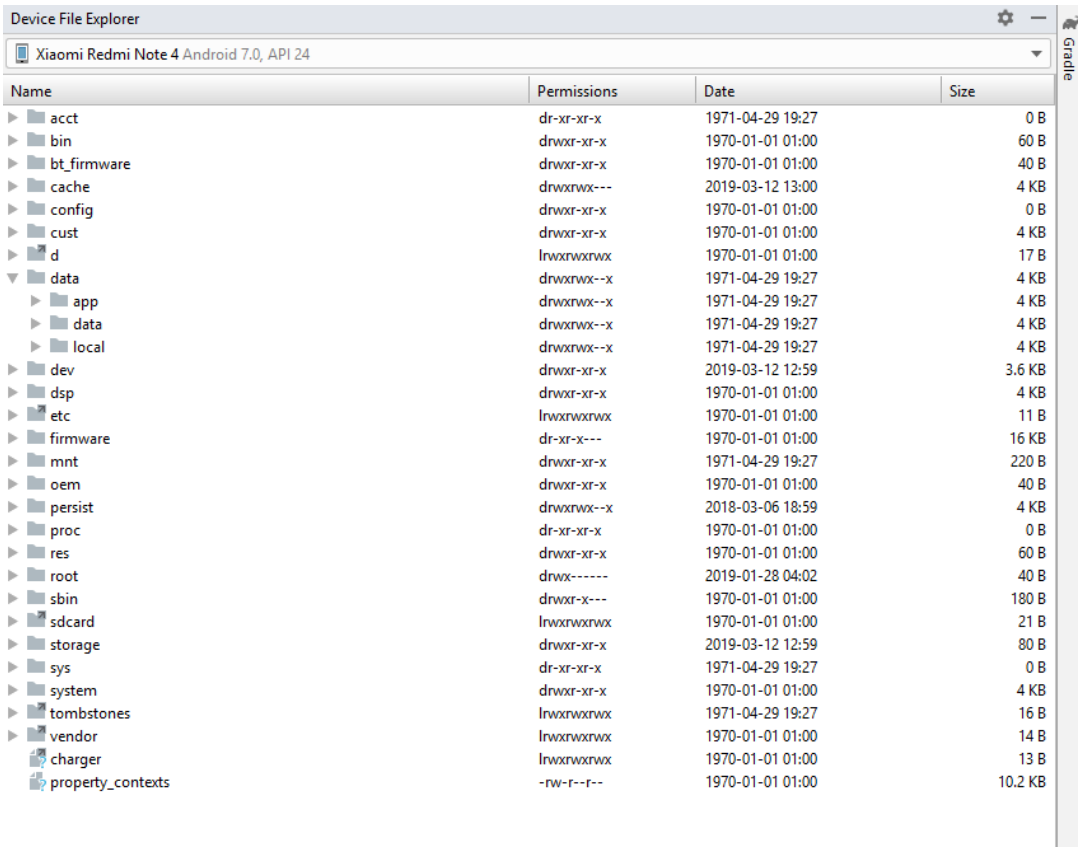


FIGURE 3.5: Internal Storage of a Mobile Device as seen from Android SDK's ADB

be done individually and physically (by meeting with all users of the application at the end of their respective studies). This was very time-consuming. Therefore, approximately during the mid-phase of the study, other design mechanisms were leveraged, that allowed the possibility of sending the user's internal database by email to the application's developer. However, due to the high complexity of the implementation of this mechanism, this design idea was deprecated, as the application had a simple purpose, and the complexity of this task was disproportionate to the rest of the application's features. The process consisted of extracting, with the help of Android's SDK (Source Development Kit), more specifically, ADB (Android Debug Bridge) the database via a USB serial port connection between the user's mobile device and a computer with Android SDK installed on it. When connected, and giving additional configurable mobile device permissions (developer permissions), this USB connection was allowed. The set of permissions that need to be configured on the device of the user will be further discussed in Section 3.4.2. In Figure 3.5 we have an illustration of the menu of the internal storage of a test mobile device. Additionally, we can observe the presence of the database inside the mobile device's storage in Figure 3.6.

Finally, by accessing the data folder in the internal storage of the device, more specifically, this path: */data/data/<packagename>/databases/<databasename>*. The database could be extracted and saved into the corresponding PC, in which data would be stored in its corresponding category of the user and stored confidentially in an encrypted USB stick. We will talk more about how the security of the confidential information provided by the users was handled in Section 3.5. Note that personal

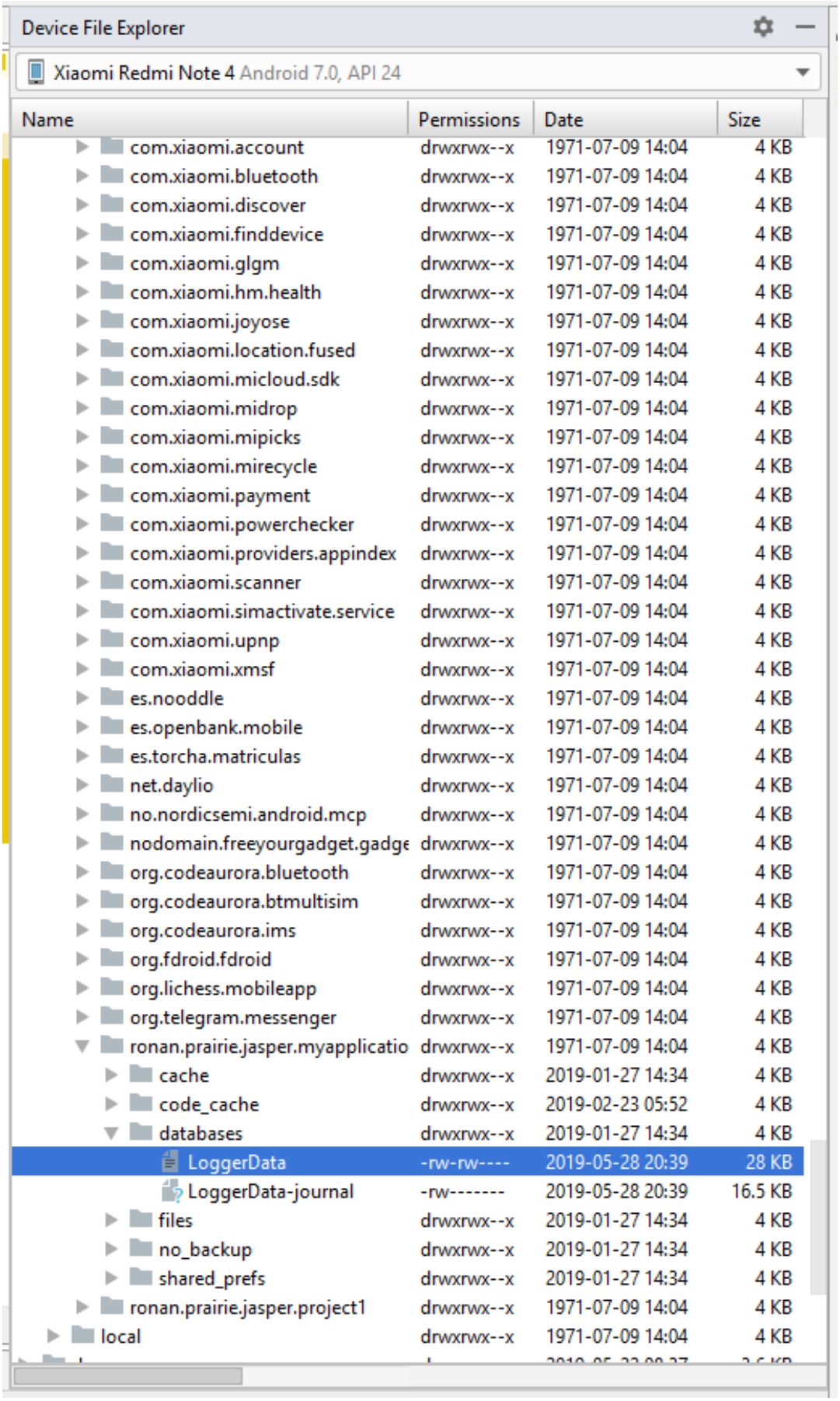


FIGURE 3.6: Database of the Android application visible in the directory hierarchy of the Android application inside the mobile device. The mobile device is connected via USB to the computer where Android SDK's ADB is running.

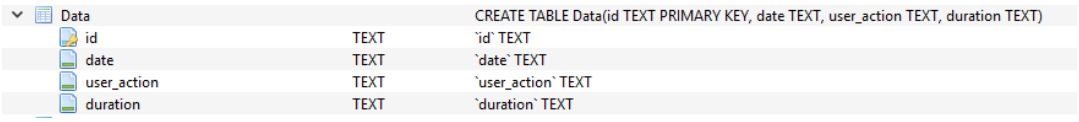


FIGURE 3.7: The only table in the database and its structure can be observed here.

information about other applications could be accessed. However, since all Android applications that are found in the Google Play store have a special mode, called **release mode** (they do not allow their information to be extracted by default), the information of other applications could not be extracted, allowing users to preserve the privacy of their mobile devices with the exception of the Android application. Note that this Android application had to be developed and then released with **debug mode** in order to allow data to be extracted. Even though a free version of the application can be found in the Google Play Store in release mode, and the initial purpose of the distribution of the applications was to give users this *.apk* file through Google Play, it was not possible in the end; and the distribution of this application had to be done individually through a shared link in Google Drive. However, a more sophisticated implementation that did not need to access the database through a serial port USB connection would require the application to have **release mode** enabled.

3.4.1 Android Database Structure

In this subsection, we will explain the structure of the own-developed SQLite3 database used in the Android application. This database was created with the purpose of storing user data in the Medical Logger application, and then extract it in the ways explained in Section 3.7. The database consists of only one table which stores all data. Figure 3.7 illustrates the structure of the only table present in the database.

In Figure 3.8 we can observe some sample data extracted from a mobile device used for testing. As we can observe, the precision of the date where the user presses the *confirm* button (which saves this data in the mobile device's internal storage according to the mobile's local time) will allow us to correlate all data from the Xiaomi Mi Band 2 device (in Chapter 4) with the type of activity the user indicated. In case the user indicated an indeterminate duration, the activity was considered to last until the next activity was marked in the Android application. This was something important to consider, as this could incorrectly categorize the heart rate of a user with a specific activity, if the user forgot to introduce an activity.

3.4.2 Developer Permissions and Requirements

The Android application was tested with an emulated device including Android API 23. In Figure 3.9 we can observe how many devices will be able to execute the Android application based on the API version. However, this does not constitute the minimum API level that a mobile device needs to have in order to execute the application. In fact, as the application does not use any special gadget or addon provided by these API levels, it is estimated to work with the minimal API level, meaning, Android Jelly Bean (4.1), which would allow the application to be executed in approximately 99.6% of devices. In the *build.gradle* field, the minimum SDK level is specified. In case of the Android application, it is API level 16.

id	date	user_action	duration
	Filter	Filter	Filter
24	2019-02-03 14:29:41	User action: caminar	Duration: No lo sé
25	2019-02-06 13:53:01	User action: ejercicio fisico	Duration: 10 horas 0 minutos
26	2019-02-06 20:30:28	User action: dormir	Duration: No lo sé
27	2019-02-19 12:16:00	User action: dormir	Duration: 4 horas 0 minutos
28	2019-02-19 15:14:00	User action: reposo	Duration: 2 hours
29	2019-03-17 19:14:33	User action: reposo	Duration: 20 minutos
30	2019-03-17 19:45:03	User action: ejercicio fisico	Duration: 1 hora 10 minutos

FIGURE 3.8: Sample data from the SQLite3 database, from a user's test mobile device, after extracting it from the user's device with Android ADB, as explained in Section 3.7.

There are a set of permissions that needed to be activated in each mobile device in order for the application to allow the USB serial port connection with a PC. Without this connection, the data study would have been impossible to perform, as information would not have been able to be extracted.

The set of permissions that are required, in a typical modern mobile device (even though developer menus varied from every Android version and manufacturer), are the following:

- Debugging - USB Debugging: enables debug mode when the mobile device is connected to an USB serial port.
- Install via USB - This feature allows installing apps through USB. It was useful through the testing phases of the application, not in the testing phase, to install the application immediately from the computer, without needing to send the application by email, Google Drive or any other form, which would take much more time. It allowed debugging the application in real time, while looking at all debug messages in the PC through Android Studio.
- USB Debugging (Security Settings). It allows granting permissions and simulating input via USB Debugging (which is often used for clicking and navigating through the application using Android Studio. This was used in the development phase of the application too.
- Verify apps via USB. It checks if apps installed via ADB are harmful. This had to be disabled, as some mobile devices restricted the installation of the Logger application without this option disabled.

This developer options menu is initially disabled and hidden to the user. In order to activate it and make it visible, the IMEI of the mobile phone (typically, this is what needs to be clicked) needs to be located and clicked a given number of times (typically, 5 times). For security reasons, the menu is not public in a factory-reset mobile devices, as its use is intended only for experts. Wrongly modifying information in this menu could cause severe consequences to the mobile device permanently. Inside

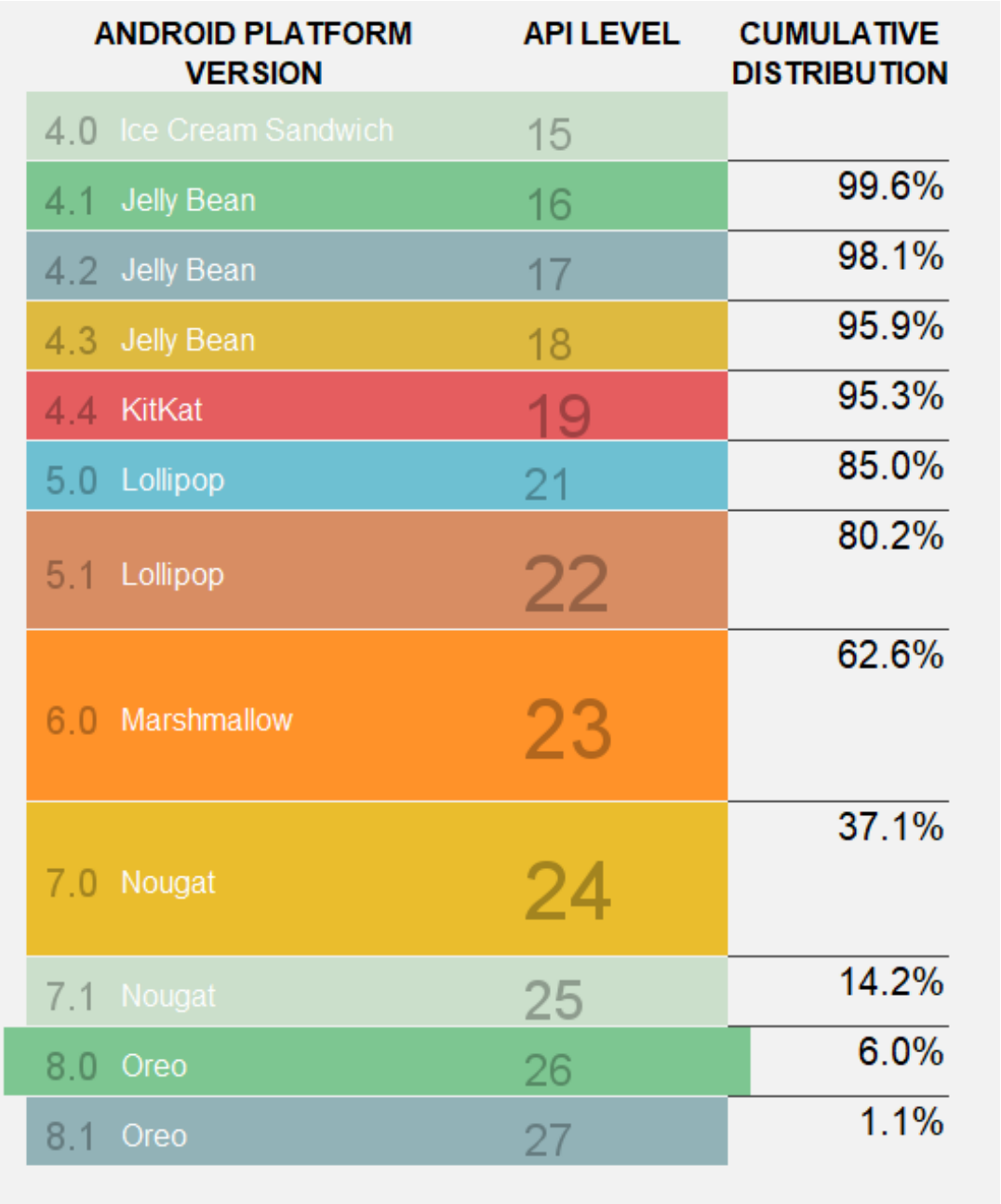


FIGURE 3.9: Here, we can observe all API levels and the estimated availability percentages in the devices throughout the world.

this menu, we can find several security options that initially restrict mobile devices to use USB Debugging and other tools that were required for extracting the database. These had to be activated, with consent from the user, and after the end of the study, disabled again to its original state, to protect users from unauthorized and further use of these tools against their mobile devices. We can also see an illustration of this menu in Figure 3.10.

As mentioned before, the corresponding set of required permissions were activated and the study could then begin, sending a Google Drive link to the user with the *.apk* file corresponding to the Logger Android application. After installing it, users would then sign the corresponding paperwork (see Section 3.5) and begin the study.

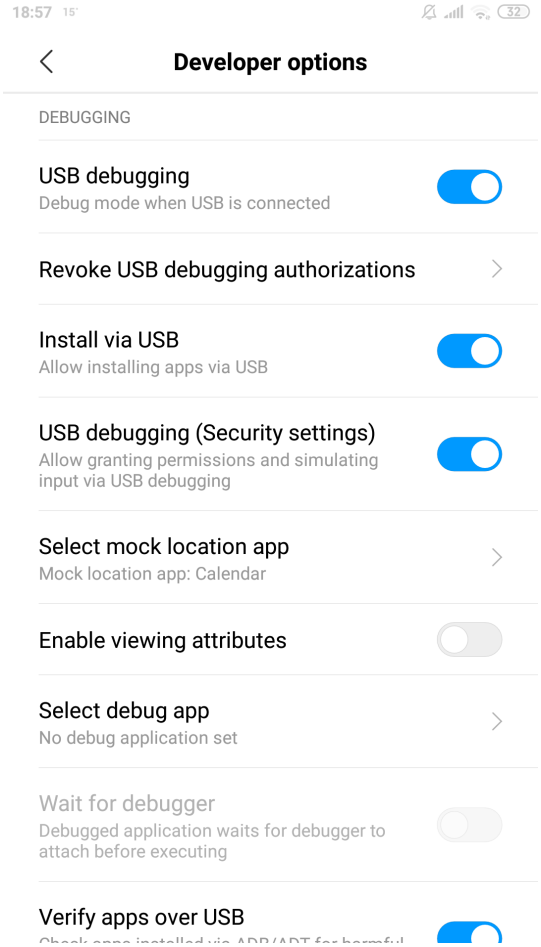


FIGURE 3.10: Android Developer Options Menu

3.5 Study Forms

In this section, the set of forms used in the study are described. The depictions of these study forms can be found in Appendix A. These forms cover consent forms, used to collect information from the user and to inform the user of the characteristics of the study; and user satisfaction forms, given after the study to see the degree of compliance with the study and satisfaction of test users.

3.5.1 Consent Form and User Profile

Users were given, prior to the study, a consent form, indicating the following information:

- Whether the user is employed
- How many times the user exercises a week
- Whether the user suffers from a congenital heart disease
- Whether the user's relatives have suffered from heart attacks in the past
- Whether the user smokes regularly or on a daily basis
- How old the user is

- The gender of the user
- The user's signature

The consent form also explained users the confidentiality elements leveraged in the study, as well as a brief description of the study and what is required of them. Also, some personal contact information was present, in case issues occurred during the study, and ultimately, to return the Xiaomi Mi Band 2 devices and meet up to extract the database information.

The template used, both in English and Spanish languages (they were given respectively, given their mother tongue), as the consent forms, can be observed in Appendix A: (Section A.1 and Section A.2). After signing and consenting to all the terms in the form, the study began (if the user met all the requirements). They would also be asked, as the consent form shows, to report any inconvenience during the study. One example would be running out of battery in the wearable device, as the chargers of the wearable devices were not provided to test users during the study.

The set of requirements of the user, prior to accepting them in the study, would exclude any user with a congenital heart disease, because it was initially thought that this could cause disruption to the normal functioning of a heart in healthy conditions. Additionally, users with congenital heart diseases were discarded because the bachelor's work is considered as a proof of concept, and we desired all our data to be as normal as possible. Also, after having a given number of participants from an age range (young, middle-aged or old), no more users from this age range would be accepted, as the study pretends to have at least 10 users for each age group; which would allow the three groups to be compared statistically in areas, such as obtaining an average basal heart rate for each one of the three groups, and later categorize test users of the real-time application to fit their needs better than having an only group.

3.5.2 User Satisfaction Form

Finally, after the three-day period of the study, users would be asked again, in another form, to express their thoughts about the usability of the Android application, as well as some personal opinions about the design of the application. This form is called the *User Satisfaction* form, which can be seen in both Spanish and English languages, in Appendix A: (Section A.4 and Section A.3).

After the statistical study, users were able to express their thoughts and ideas about the usability of the application, according to all initial requirements of the application, which can be observed in Section 3.2. A compilation of all these issues is explained in Section 3.9.

3.6 Issues

During the extraction of data of the Android application (not considering issues related with Xiaomi Mi Band 2 devices, which will be further discussed in Chapter 4), there were some issues that need to be addressed:

- Test users with iOS mobile devices were not eligible to participate in the study, for compatibility reasons (the application was developed only for Android).
- Users with Samsung devices, after the period of their study, showed that the database could not be extracted as in any other mobile devices. After further

investigation, it was shown that Samsung devices are incapable of executing the *run-as* command, which makes USB Debugging in Samsung devices completely void (this was caused by Samsung's mobile design, as they dropped a *setuid* flag, making *run-as* not able to switch to a different uid). Therefore, debuggable applications can not exist in Samsung devices. Due to this issue, which was discovered after three users had already performed the study, all the information had to be discarded and new users had to be found to replace them.

- Users that had initially tried to perform the study with the Google Play application (which, as mentioned before, did not have *debug mode* enabled, but *release mode*) had to be dismissed because the database could not be retrieved.
- When extracting data, some devices had a smaller number of samples than others. This raised an alarm and required further investigation. The issue was that, some users, unbeknownst of the repercussions of connecting the Xiaomi Mi Band 2 devices to their mobile devices via Bluetooth, had misconfigured the Xiaomi devices. Since GadgetBridge allowed setting a heart rate measurement interval of 1 minute, and the standard Mi Fit vendor application allowed a higher range, when users connected the mobile devices with the Xiaomi device via this application, the standard heart rate measurement interval was increased, causing measurement intervals to be every 3 or 5 minutes instead. Therefore, the set of users that did this (a total of three users) had to be dismissed from the study and new users were found to replace their data.

Due to all these issues, the data collection process ended up being much longer than expected.

3.7 Data Extraction

In this section, the process of extracting data from a Xiaomi Mi Band 2 device into a mobile device is explained. The main tool used for the extraction, as the official application of Xiaomi, Mi Fit, does not allow for the inspection of these databases (is protected by default and it is impossible to access this data), a non-official open-source application, called GadgetBridge, has been used instead.
GadgetBridge is an application that allows any mobile device, with an Android version higher than v4.4, to use several devices as a standard replacement for the original vendor's closed-source application, and also without the need of creating a Xiaomi (or any other vendors) account, allowing these vendors to store and have a free pass to all information stored in these devices. The list of supported devices is numerous, but for the purpose of this bachelor's work, knowing that all Xiaomi Mi Band versions are supported (1, 2, and 3) is enough.

Aside from the normal features that any vendor application has, it allows directly downloading the database from the Xiaomi Mi Band 2, through Bluetooth, to a mobile device, and then export this database to anywhere - Google Drive is the technique used to extract the database and keep it updated. This feature was especially useful, as data needed to be extracted and analyzed efficiently. This is the main reason why GadgetBridge was chosen to be the ideal application for the data extraction process. The GadgetBridge application has, however, a limitation: the minimum heart rate measurement frequenty that can be established and configured

to a Xiaomi wearable device is one minute, and can not be diminished any further.

When a database is extracted into the mobile device, and because there are several wearable devices that could be connected to the same mobile device, the database structure, as seen in Section 3.8, Figure 3.15, there is a field that unequivocally identifies the device it has introduced the data, called *DEVICE_ID*. For example, imagine a user had three different devices: a Xiaomi Mi Band 2, another Xiaomi Mi Band 3, and finally an Apple Watch 2, and decided to use all three devices for different tasks throughout the day. Each device would be internally stored with a distinct DEVICE_ID.

Thus, the same mobile device was used for extracting data from the Xiaomi wearable devices, and one final database file was created, with the union of data for all three devices. After having the database periodically uploaded to Google Drive, further analysis could be made. This analysis is made in Section 4.1.

3.8 Xiaomi Database Structure

In this section, a comprehensive description of the Xiaomi Mi Band 2's database structure is made. The structure of the database is proprietary and created by Xiaomi. However, GadgetBridge's application joins multiple database structures into a joint one with all data inside. In this database, all information is stored temporarily in the local cache of the Xiaomi Mi Band 2 device, until connected through Bluetooth to a mobile device. This data consists of the heart rate measurements of the user, and must not be confused with the data collected by the Android application, which was already explained in Section 3.4.1. Information is retrieved by the mobile device and the information is deleted from the local cache of the Xiaomi device. This way, the storage of the Xiaomi device doesn't require to be too big, as it only needs to temporarily store information. A possible disadvantage of this is that, when the local storage of the Xiaomi device becomes full, information is overwritten and data that has not been backed up is lost. However, this has not happened during the duration of the whole statistical study.
On the other hand, the local storage capabilities of the Xiaomi device do not have any repercussion when it comes to the integration of the system with a social robot in real time, as heart rate data will be transmitted via Bluetooth to the robot, meaning that local storage will not be used at all.

The MySQL Workbench E/R (Entity-Relation) database schema for the GadgetBridge database can be observed in Figures 3.11 and 3.12. Remember that the GadgetBridge database is the union of multiple Xiaomi devices that have been synchronized via Bluetooth with the mobile device where GadgetBridge is installed.

Having a look at the database information, whose study has been done utilizing the DB Browser for SQLite application for the Windows 10 Operating System, we can observe several data. All this data is relevant to the application. First, there is a table called **DEVICE**. This table is important because it allows us to determine a relationship where data is, and which device has produced this data. The table contains the following information:

- Row ID, represented by the field *_id*, to avoid row duplicates as entries in the database.

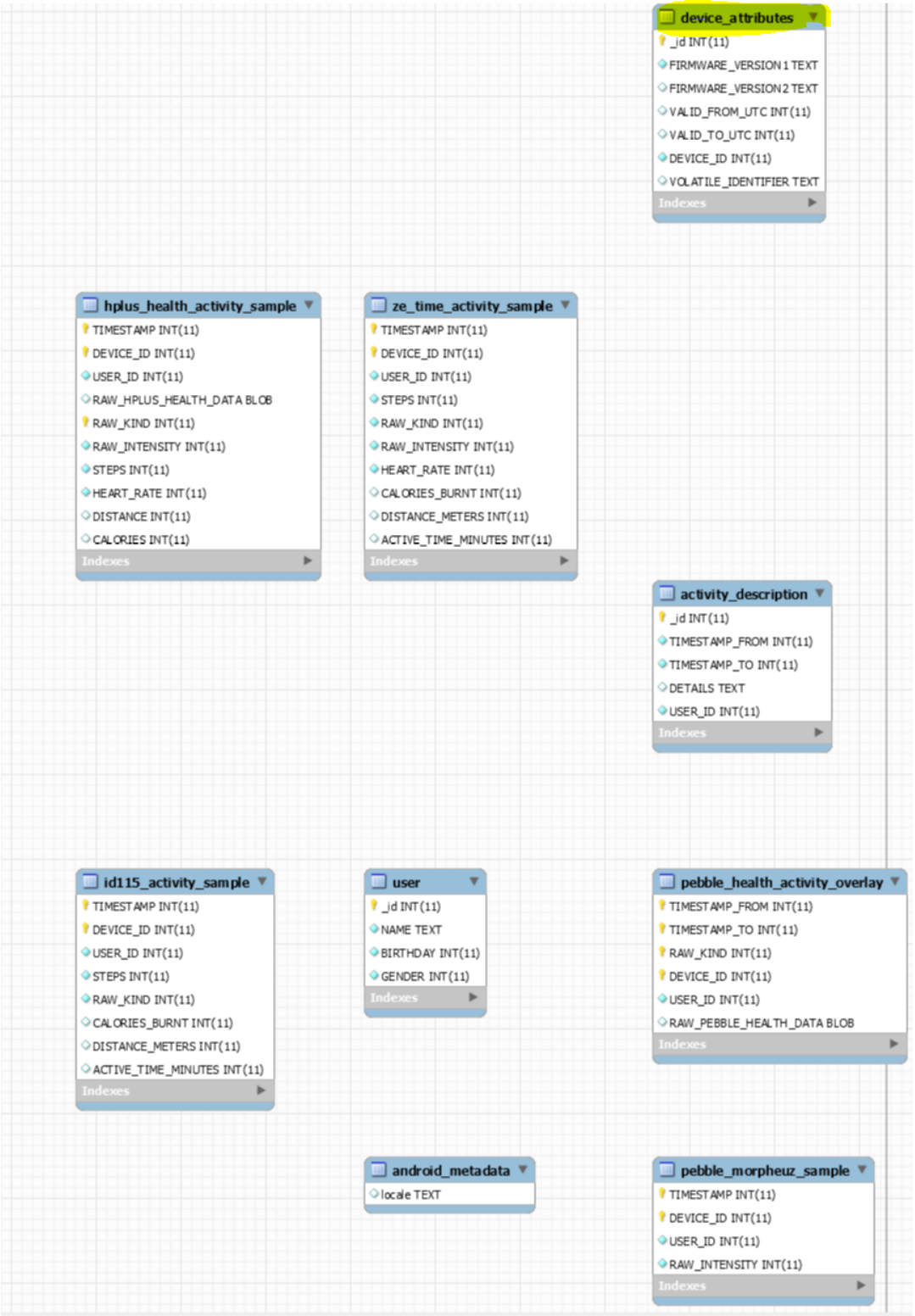


FIGURE 3.11: Xiaomi Mi Band 2's database schema.

- Device Name, represented by the field *NAME*. This is always MI Band 2.
- Device Manufacturer, represented by the field *MANUFACTURER*. This is always **Huami**.
- Device Identifier, represented by the field *IDENTIFIER*, which is especially

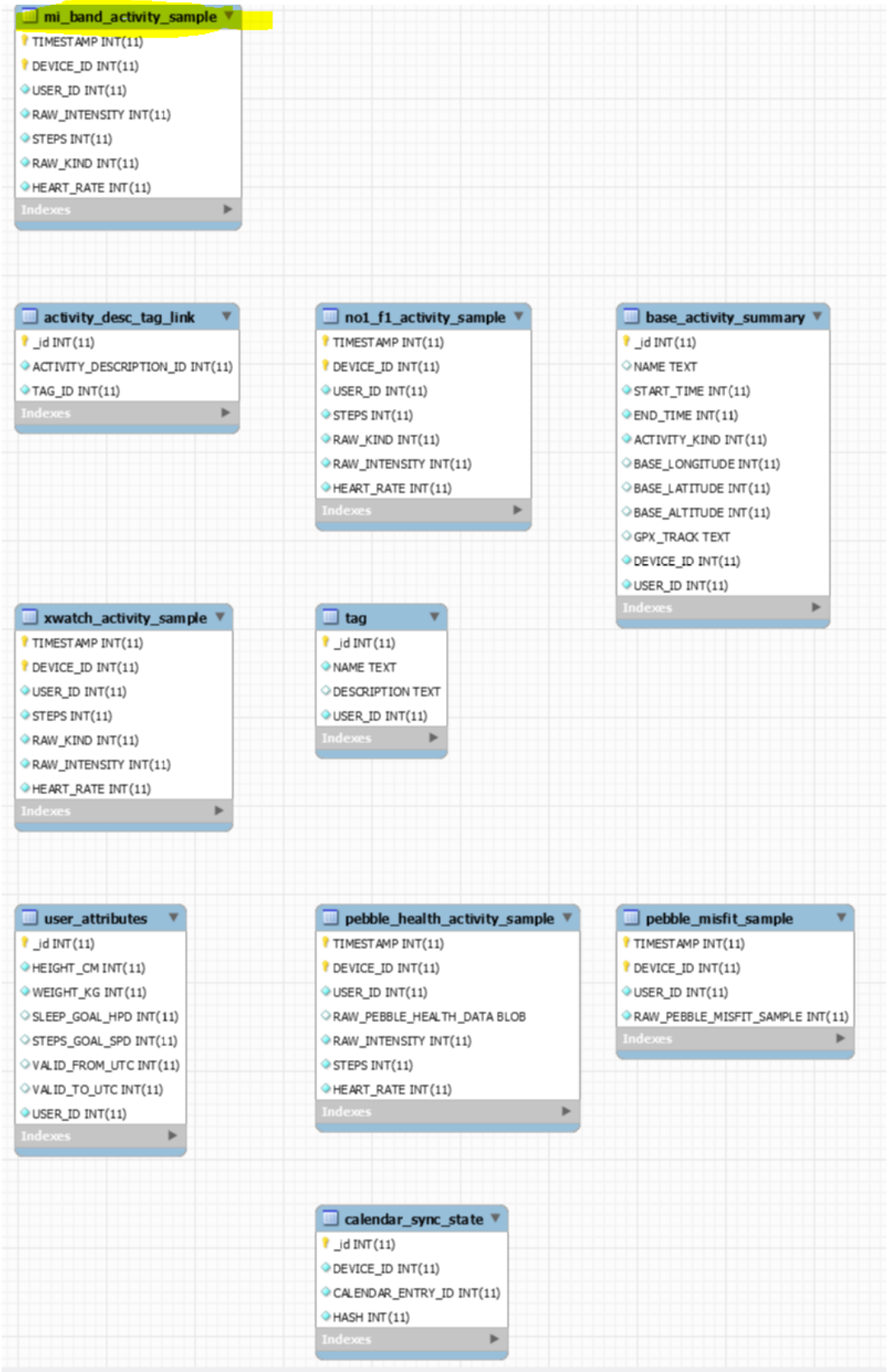


FIGURE 3.12: Xiaomi Mi Band 2's database schema.

useful in this case. This row represents the Bluetooth MAC Address previously addressed. This will allow, when having a database containing information from several devices, such as this case (in which data from three different

	_id	NAME	MANUFACTURER	IDENTIFIER	TYPE	MODEL
	Filter	Filter	Filter	Filter	Filter	Filter
1	1	MI Band 2	Huami	ED:61:38:85:D1:FC	11	V0.1.3.3
2	2	MI Band 2	Huami	D8:59:59:8A:E5:69	11	V0.1.3.3
3	3	MI Band 2	Huami	E6:A1:BD:94:31:54	11	V0.1.3.3

FIGURE 3.13: Data present in Device table.

Table: DEVICE_ATTRIBUTES

	_id	FIRMWARE_VERSION1	FIRMWARE_VERSION2	VALID_FROM_UTC	VALID_TO_UTC	DEVICE_ID	VOLATILE_IDENTIFIER
	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	1.0.1.81	NULL	1548616774480	NULL	1	NULL
2	2	1.0.1.69	NULL	1548617016700	NULL	2	NULL
3	3	1.0.1.69	NULL	1548967761139	NULL	3	NULL

FIGURE 3.14: Data present in Device Attributes table.

devices is mixed. This will be explained in Section 3.7), filtering all the information to study data from one device at a time.

- Type of Device, represented by the field *TYPE*, always 11, but this information is irrelevant to us.

- Firmware Model, represented by the field *MODEL*, which is also especially interesting, as previously mentioned in Chapter 2, because we can differentiate the firmware model of the device. With this firmware information, the specific capabilities of the hardware device can be observed. This had to be checked after purchasing all three Xiaomi Mi Band 2 devices, to make sure that these devices were able to support all the requirements for the study. In all three devices, the firmware version is v0.1.3.3, which corresponds, as previously mentioned in Section 2.2.5 ([25]), to the current newest firmware version. Heart rate measures are available, as well as accelerometer measurements and all other features (notifications and activity recognition).

All information contained in this table can be observed in Figure 3.13.

Additional device information resulting from another table can be seen here. This device information is called *DEVICE_ATTRIBUTES*. It allows checking the old format for describing the device's firmware (1.0.x.x), also called as the original Mi Band 2's firmware version ([25]):

In Table 3.8, what initially seems like the manufacturing date of the device, can be observed. However, after further investigation, this is the last time the device ran out of battery; in the field *VALID_FROM_UTC*, which is in a standard UNIX epoch time (which corresponds to the amount of seconds elapsed from January 1st, 1970 at 0.00 AM). A simple conversion can be made to check the last time these devices ran out of battery, which is, to this date, about 2 months ago. This data can be then compared with the data present in Figure 3.14 to check the last charging time of the device. This is very important, because we can identify errors in the Xiaomi device (such as synchronization errors, or having the Xiaomi device displaying an incorrect time because it ran out of battery and did not synchronize again with a mobile device) by looking at the *DEVICE_ATTRIBUTES* table.

- Device 1: 1548616774480: Sunday, January 27, 2019 8:19:34.480 PM GMT+01:00

	TIMESTAMP	DEVICE_ID	USER_ID	RAW_INTENSITY	STEPS	RAW_KIND	HEART_RATE
	Filter	Filter	Filter	Filter	Filter	Filter	Filter
61123	1548831000	2	1	54	0	112	73
61124	1548834480	3	1	26	0	112	54
61125	1548834540	3	1	52	0	112	83
61126	1548834660	3	1	0	0	112	54
61127	1548834840	3	1	25	0	112	64
61128	1548834900	3	1	6	0	112	48
61129	1548838560	3	1	0	0	112	48
61130	1548838680	3	1	74	0	112	65
61131	1548848640	2	1	11	0	112	84
61132	1548848700	2	1	0	0	112	82
61133	1548848760	2	1	1	0	112	85
61134	1548849000	2	1	9	0	112	56
61135	1548849120	2	1	0	0	112	76
61136	1548849180	2	1	0	0	112	76
61137	1548849240	2	1	0	0	112	74
61138	1548849300	2	1	0	0	112	84
61139	1548849360	2	1	0	0	112	73
61140	1548849420	2	1	0	0	112	75
61141	1548849480	2	1	0	0	112	78
61142	1548849540	2	1	0	0	112	84

FIGURE 3.15: Data present in Mi Band Activity Sample table. This is one of the tables of the GadgetBridge database, where we can observe data from all three devices is mixed up in the same table.

- Device 2: 1548617016700: Sunday, January 27, 2019 8:23:36.700 PM GMT+01:00
- Device 3: 1548967761139: Thursday, January 31, 2019 9:49:21.139 PM GMT+01:00

As devices are recharged personally at the same time, and are not unloaded from the USB serial port until all three devices are fully loaded, this information is useful when being analyzed, which begs the question: does Device 3 have more battery capabilities than the other two devices?

As shown here, there are two devices with the firmware version 1.0.1.69 (*D8:59: 59:8A: 59:69* or Device 2; *E6:A1: BD:94: 31:54* or Device 3) and one device with the firmware version 1.0.1.81 (*ED:61: 38:85: D1:FC* or Device 1). Referring back to [25], device features can be checked. Firmware version 1.0.1.69 is observed to have been previously tested with the data extraction software explained in Section 3.7; and firmware version 1.0.1.81 has not. So, keeping that in mind, the database is is further analyzed, taking special precautions when performing data extraction operations in Section 3.7 for Device 1.

Note that, a better suggested design for relating both the **DEVICE** and **DEVICE_ATTRIBUTES** tables, as having related tables without a foreign key seems inefficient, would be to implement a foreign key that references the device identifier of table **DEVICE** to the identifier of table **DEVICE_ATTRIBUTES**.

Finally, the table that contains all heart rate and accelerometer measurements is contained in the table **MI_BAND_ACTIVITY_SAMPLE**.

We can see some data included in this table here:

In this table, we can observe the following useful information:

1. Device ID, either 1, 2 or 3, which can be traced back - and ideally through a foreign key - to the **DEVICE** table, represented by the field *DEVICE_ID*
2. Heart rate of the user at the calculated time, represented by the field *HEART_RATE*
3. Timestamp (in UNIX Epoch time format, represented by the field *TIMESTAMP*, which is later useful to graph the heart rate of the user at a specific given time, and plot all the results from a user in a graphical and elegant way.
4. Accelerometer values corresponding to the accumulated amount in the event of measurement, represented by the field *RAW_INTENSITY*.
5. A hard to comprehend field, called *RAW_KIND*. This field required lots of code reviewing to check the actual meaning of the values in this column. After a thorough investigation, it was discovered that this field can vary depending on the firmware version. It corresponds to some constants, and the hardware of Xiaomi Mi Band 2 automatically tries to categorize the set of activities performed during the day. There are some applications that try to improve this system by actively asking the user which kinds of activity it they are making, such as *Mi Fit*, the official Xiaomi application for wearable devices.
   For example, in firmware version 1.0.1.50, there are some values that can be extracted, by looking at own data and comparing values:
   - 1: walking. Also, codes 16, 17 and 26 (rarely) also appear when walking.
   - 50, 66, 98: running.
   - 82: beginning and ending of running.
   - 112: deep sleeping, which is set to this value when the raw_intensity value is 0.
   - 122: standard sleeping, which is set to this value when the raw_intensity value is smaller than 20.
   - 17: 1 minute before starting to talk and 1 minute after walking.
   - 28, 105: waking up.
   - 106, 123: falling asleep.
   - 80: when the number of steps is 0.
   - 89: when the user wakes up.

   Of course, these categorizations are completely different between firmware versions, so it is very difficult to analyze based on this data. Also, the hardware classifier may fail, so this field will ultimately not be used for data analysis. However, it is interesting to observe what this constants represent, which is an attempt of cataloging every action of the user locally within the device.

All the other tables were not included in this analysis, as they were either completely empty or they did not contain any relevant data worthy of analyzing.

Data Storage Efficiency

With regards to the data storage efficiency of the Xiaomi database, and having previously analyzed Xiaomi's database structure, a pessimistic prediction about the amounts of data the local cache of the Xiaomi Mi Band 2 device needs to store is made. This, of course, depends on the amount of samples taken and the frequency

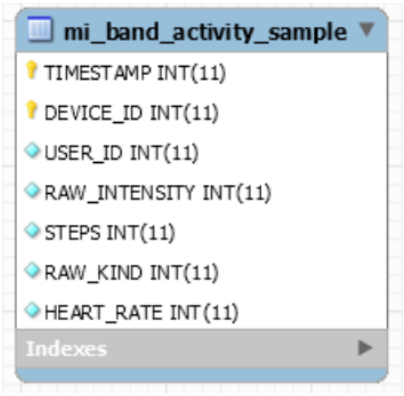


FIGURE 3.16: Table structure of **MI_BAND_ACTIVITY_SAMPLE**.

of these samples. For this prediction. This is the lowest available heart rate measurement frequency that the GadgetBridge application allows Xiaomi Mi Band 2 devices to be configured to. We can see a depiction of this by looking at the structure of the *MI_BAND_ACTIVITY_SAMPLE* table (Figure 3.15) when this interval is set. For example, having UNIX epoch timestamps from rows 61135 to 61137 all represent an increment of 60 after the previous one: 1548849120, 1548849180, 1548849240. Note that there is a missing value between row 61134 and row 61135. These types of internal hardware errors are unpredictable and we could not do anything about it, after analyzing possible causes of the error.

A high percentage of all data is present in the *MI_BAND_ACTIVITY_SAMPLE* table, as many tables have no information or just a couple of rows with no more than 100 bytes each. So, in order to make a prediction about the practical size of the local storage, this table must be considered as the main and only source of data, neglecting in the initial calculations all other table sizes; and adding these sizes from all other tables in the end. When calculating the size of all other tables, the size is not greater than 500 bytes.

As observable in the database scheme in Section 3.8, Figures 3.11 and 3.12, the following table structure is found:

As all elements are integers, and integers are represented as 4 or 8-byte numbers, we need to know the size of integers. In most database engines, it is considered as a 32-bit number, but since the worst-case scenario is considered here, it will be assumed that the integer is represented as a 64-bit number, and therefore, stored as an 8-byte integer. The sum of all fields, is 7 columns, all of them represented by integers. Therefore, the size of a row is 7 columns, multiplied by 8 bytes (the size of the integer) per column. This results in a total of **56** bytes to represent one row of data. Now, considering that one row is stored into local cache every minute (one row per heart rate measurement) until downloaded into a mobile device, a total of 56 bytes is stored every minute. This is equivalent to 3.360 bytes every hour, 80.640 bytes every day, and 2.419.200 bytes every month. So, approximately *2.3 MiB* are needed to store one sample every minute, for the whole duration of a month. Therefore, based on these calculations, since the current capabilities of storage and memory are much higher than this, there should not be any issue to store data for months, or even years, without overwriting data or losing it - and keeping this data locally.

Therefore, as a conclusion, not downloading this data into mobile devices (as a way not to overwrite cached data) fast enough will not be a concern for the whole

purpose of this work.

3.9 Summary

A compilation of all issues mentioned by users in the user satisfaction form was made. These issues were described in the user satisfaction (see Section 3.5.2) The following list summarizes all of these issues or suggestions on how to improve the application:

- Have more options other than the five already pre-defined buttons for determining more specific activities.
- Make the application available on the iOS platform, in order to attract other possible test users with a non-compatible Operating System in their mobile devices. That means, develop an *platform-independent* application
- Implement an *alarm* or a vibration in order to remind users to note their activities. This was the idea that most users requested after the study.
- Introduce more dynamic icons, being these more creative. Also, have a more colorful UI design, and include more images in the screen. However, as seen in Section 3.2, this was not the purpose of the Android application.

The application had an average usability value of *8.0* according to users.

Finally, users noted the set of activities that they made with an average frequency of 8 to 10 times per day (8.5 times every day), having zero users with a lower average amount of activities noted than 4 per day, and being the highest contributor a user with an average of 18 activities per day.

With regards to the analysis made to the Xiaomi database, public documentation about the structure and format of the data and databases was *completely non-existent*. Therefore, making an analysis of the internal structure of the Xiaomi device was necessary, to make sure that the capabilities of the device would be enough to develop this bachelor's work.

We conclude that the storage capacity of the Xiaomi wearable device's local cache is enough to store data in devices for more than three days. Therefore, if further analysis or study is made, connecting the Xiaomi device with a mobile device via Bluetooth every three days or less will be enough

Chapter 4

Data Analysis

This chapter focuses on the analysis and correlation of all data involved in the statistical study. Both data from the Android application, as well as data obtained from the Xiaomi wearable devices, is involved in this data analysis process. We will attempt to produce conclusions that can be used for a better understanding of our statistical test users, and possibly use this information for developing Machine Learning predictions, as previously explained in Section 1.3.

4.1 Data Analysis

In this section, a comprehensive data analysis of all the data extracted from Chapter 3 is made. For this, we need to consider several categories. First, we will extract global information from users, obtaining a general overview of the statistics of all users. Then, going into a higher level of detail, we will make more specific comparisons, related to the age and gender of test users. As it is yet not clear whether the algorithm to develop must consider user-specific data as input (meaning that, when a user launches the final real-time application, some parameters regarding this user will be loaded), or simply use a generalized formula for all users, this study will be made assuming the most complicated case. That is, we will include parameterized data from users.

The information from which we make the statistical study, is, in the end, extracted from a sample size of **24** users. The general study was made with a total of 36 users, however 12 out of these 36 had data that was either corrupt or invalid. These problems have been detailed in Section 3.6.

4.1.1 Initial Data Set

In this section, an explanation about the initial data set will be given. According to the data collection made, a total amount of 223.594 samples, meaning, heart rate measurements, were initially present. Most of these measurements were discarded, because they included periods where Xiaomi devices were not worn, because they were idle, or in between batches of test users. Therefore, all periods where the wearable devices were not quantifiable, were removed. After this initial filtering, a total of **30.695** samples were left. As the Xiaomi Mi Band 2 device is not equipped with a sensor that indicates whether a user is wearing the device, the removal process automatically took all heart rate values out when they were 0 or 255 (these two values appeared when the sensor did not catch a pulse, thus, when a device was not being worn by a user).

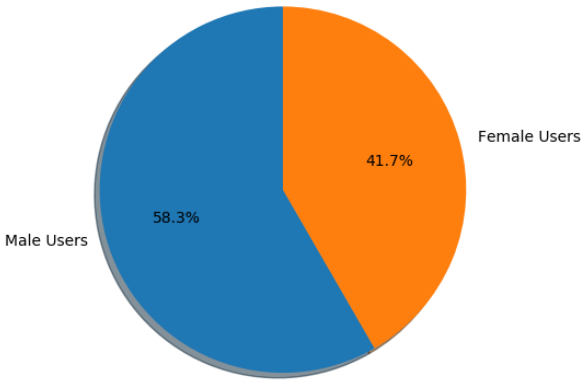


FIGURE 4.1: Gender Proportion of Statistical Study

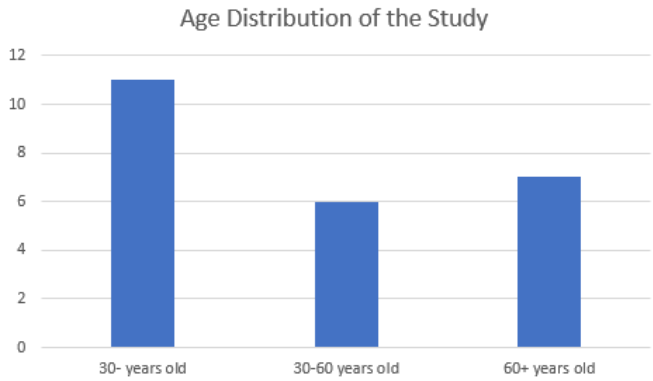


FIGURE 4.2: Age Proportion of Statistical Study

There is some information from this dataset that we shall show, in order to graphically see the features of our dataset. in Figure 4.1, we can observe that a majority, although not overwhelming, of all users, were male. Also, note that none of the female users, out of the total 10 females that participated in the data collection process, performed physical exercise. However, seven out of the 11 males that participated in the study performed physical exercise at some point during the data collection process. ALso, in Figure 4.2, we see that most users were in the first age group (less than 30 years old). It is also important to see that, even though there is a 27.1% of users in age group 2 (between 30 and 60 years old), all these users are very close to hitting their sixties. This is something that will be crucial to consider in Chapter 5 when building the Machine Learning model.

Also, we can observe the number of heart rate samples made for each age group, which will be helpful to give us an idea of how much information from each age group we possess:

After removing from the initial data set all heart rate values of 0 or 255, these data was saved in another file, containing only timestamps where users wore devices, and the corresponding heart rate values. After this, an additional cut-off had to be made from this file, removing all values exceeding a heart rate measurement over 220. This cut-off was conservative. As proven by Tanaka H., Monahan KD., Seals DR. ([29]), the maximum theoretical heart rate of a person can be calculated

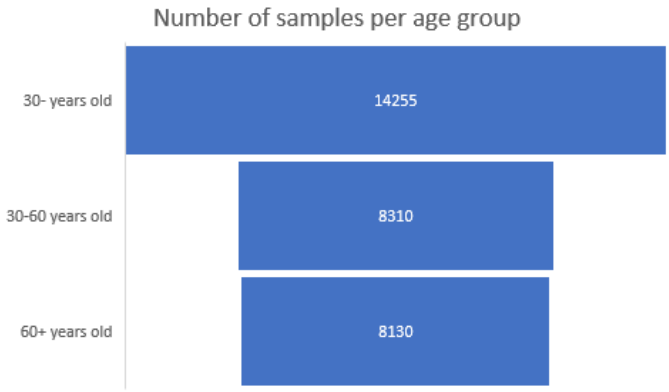


FIGURE 4.3: Percentage of measurements in age groups and their respective amount of measurements.

based on its age, never being able to surpass a value of 208. However, another scientific study ([30]) establishes that the 220-age formula can be applied to healthy individuals. As all individuals in the study are healthy, this is also taken into consideration, and both scientific studies are taken into account for removing outliers. Therefore, assuming that a user can be 0 years old at minimum, his/her heart rate will be, theoretically, 220 at maximum. Therefore, all heart rate measurements surpassing this value are removed from the data set, justifying their removal as outliers.

Additionally, after removing these values, a final outlier removal process is performed, removing all outliers that do not comply with Tukey's rule. Tukey's rule says that all outliers in an statistical study are values that are 1.5 times higher than the interquartile range from the quartiles, and considering that these quartiles are in the 25th and 75th percentile. This outlier removal process was also very conservative, as both considered quartiles ranges were 20% higher than in the original Tukey's rule.

Finally, after this initial data set filtering of outliers and non-theoretical values, the statistical study began. Note also, that after removing these values, there could be some gaps when studying dates, and these heart rate measurements were missing. To automatize the data analysis process, this was considered, and in case of a user setting a date in the Android application and later not appearing in the corresponding data set, a secondary value for this date would be taken. This secondary date would substitute the user's written-down date, and would be one minute later than the original date. For example, if a user had written down that its physical exercise activity would start at 2019-03-01 22:03:57 (March 1st 2019, 22.03.57), the data analysis algorithm would replace this date with 2019-03-01 22:03:00 (March 1st 2019, 22.03.00). Had this value not been found, then this primary date would be replaced with the secondary date 2019-03-01 22:04:00 (which is, one more minute). Note that the Xiaomi device only stored one value per minute, as explained previously in Section 3.8. There were no cases of not finding two consecutive minutes in any of the users, as expected.
Therefore, as the probability of not having two consecutive dates (having previously deleted two consecutive measurements due to their being outliers) is slim, there were no errors in the automatization of the analysis of the data set.

Categorizing all data into types of exercise, according to all values introduced by the user to the Android application and the corresponding timestamps, we obtain,

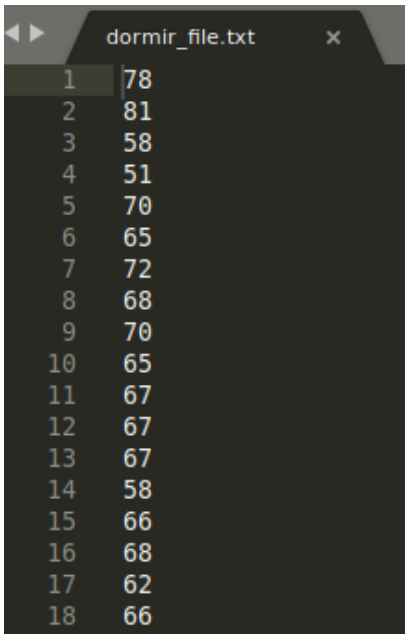


FIGURE 4.4: Example of heart rates in an indivual file, for the *sleeping* activity..

for each user, a list with all heart rates during the time when this user was in the data collection process. An example of these individual files can be found in Figure 4.4. Therefore, as all this data was separated by user, in future sections we will be able to join this data to produce data corresponding to different age groups, or different gender.

4.1.2 Global Data

From the aforementioned sample size, some statistical descriptors (*mean, median, standard deviation*) will be computed, both for all data globally, as well as from relatively-global data from all categories. This means that, for each category (sleeping, resting, exercising, walking and others), a mean, median, standard deviation and mode will be computed.

As previously shown in Section 6.1, the standard deviation seems to be very high, which is an indication that data will be disrupted. In Table 4.1, a standard deviation of about 20% can be observed.

Looking at categorical data, the following has been extracted, rounding all numbers to the nearest hundredth:

Category	Mean	Standard Deviation
Walking	78.51	18.45
Sleeping	71.57	16.87
Resting	74.06	17.93
Physical Exercise	85.46	25.58
Others	89.3	19.99

TABLE 4.1: Global Categorical Data

A visual representation of the data present in Table 4.1 can be observed in Figure 4.5.

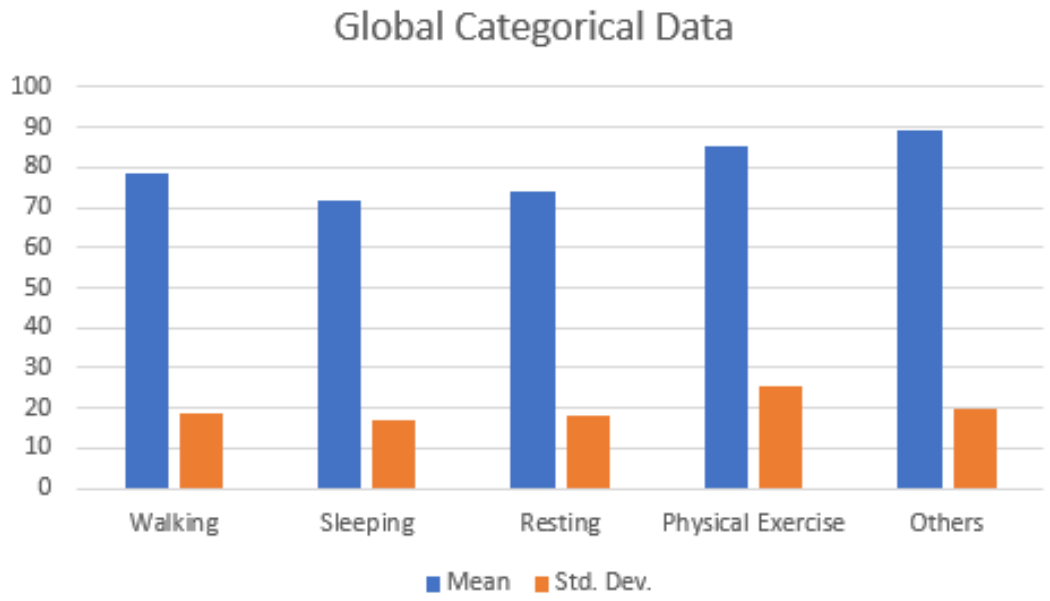


FIGURE 4.5: Visual Representation of Global Categorical Data.

Analyzing this data, a distinction can be made, primarily, between users doing physical exercise (intensive work), users walking, and users sleeping or resting. Note that activities that fall under the category *others* shall not be taken into consideration when making conclusions, as this category was reserved by users when categorizing their current activity would not fall into any of the other categories. Also, note that the standard deviation of the category *physical exercise* is much higher than any other category. Knowing the hardware specifications of the Xiaomi device, and knowing how the heart rate sensor operates, an assumption can be made that this is caused due to a high mobility of the arms and/or wrist during the physical exercise. This also depends on the type of physical exercise being made by the user: aerobic exercise, such as running, proves this theory, but there are anaerobic-type exercises, such as lifting weights, which will raise the heart rate of the user without increasing the accelerometer values of the Xiaomi device. Therefore, having a higher sample size, or distinguishing between aerobic/anaerobic exercise types would have been a better design decision when making the Android application, as it would have probably made a difference between these two categories.

Continuing the analysis, it is observed that the resting and sleeping categories have the two lowest mean values, with also similar standard deviations, when comparing them with the rest of activities. Moreover, the mean of the *resting* category, is expectedly, slightly higher than the *sleeping* category. Therefore, and having a look at this global statistical data, we can distinguish all four desired categories: sleeping, with the lowest possible heart rate and the lowest standard deviation; resting, sleeping and physical exercise, with the highest mean and standard deviation.

Finally, The lowest standard deviation, which falls under the *sleeping* category, also proves the previous analysis made, which is that it is directly influenced by the amount of wrist/arm movement during the activity. As sleeping is the activity with the lowest possible movement throughout the day, this would make sense.

4.1.3 Age comparison

In this subsection, and age comparative study will be made, analyzing three different age-range groups. In this list, the final number of users in each category is also shown:

- Age Group 1: users between 0 and 30 years old. From the sample size, a total of 11 users fall under this category (45.83%).
- Age Group 2: users between 30 and 60 years old. From the sample size, a total of 6 users fall under this category (25%).
- Age Group 3: users older than 60 years old. From the sample size, a total of 7 users fall under this category (29.17%), being the oldest user 81 years old.

In Table 4.2 we can observe the set of data that is extracted for each age group.

Age Group	Category	Mean	Standard Deviation
1	Walking	79.03	18.93
1	Sleeping	72.86	14.96
1	Resting	76.18	15.99
1	Physical Exercise	75	22.29
1	Others	82.66	17.06
2	Walking	78.81	20.49
2	Sleeping	71.57	16.87
2	Resting	74.06	17.93
2	Physical Exercise	85.46	25.58
2	Others	75.88	15.88
3	Walking	76.77	15.97
3	Sleeping	69.62	13.57
3	Resting	73.04	12.18
3	Physical Exercise	??	??
3	Others	76.23	15.03

TABLE 4.2: Categorical Data for all age groups (0-30, 30-60, 60+)

We can observe Table 4.2 in Figure 4.6.

We will now make an analysis of each one of the age groups. As group 1 is the most predominant one in the study, with a total of 11 test users, data can be more trusted. Note also, that the observations previously made about the standard deviation of the physical exercise category can be proven here, both in groups 1 and 2 (having, for their respective age groups, the highest standard deviation).

The physical exercise data from age group 3 is missing. The reason is that none of the six final test users from this age group (60 years old or older) performed any physical exercise during the study, or at least did not note it in the Android application.

Also, a difference between the mean of heart rates for each of the categories can be observed. It seems like each age group reduces their mean heart rate for each activity with respect to the previous age group. For example, walking in age group 3 has an average of 2 beats per minute less than in age groups 1 and 2. This can be caused by the natural decay of the human body over the years. It has been proven that the maximum theoretical heart rate of a person, as well as their maximal oxygen intake, is determined by the age ([29]; [31]). So, the older people get, the less beats per minute the heart is able to make.

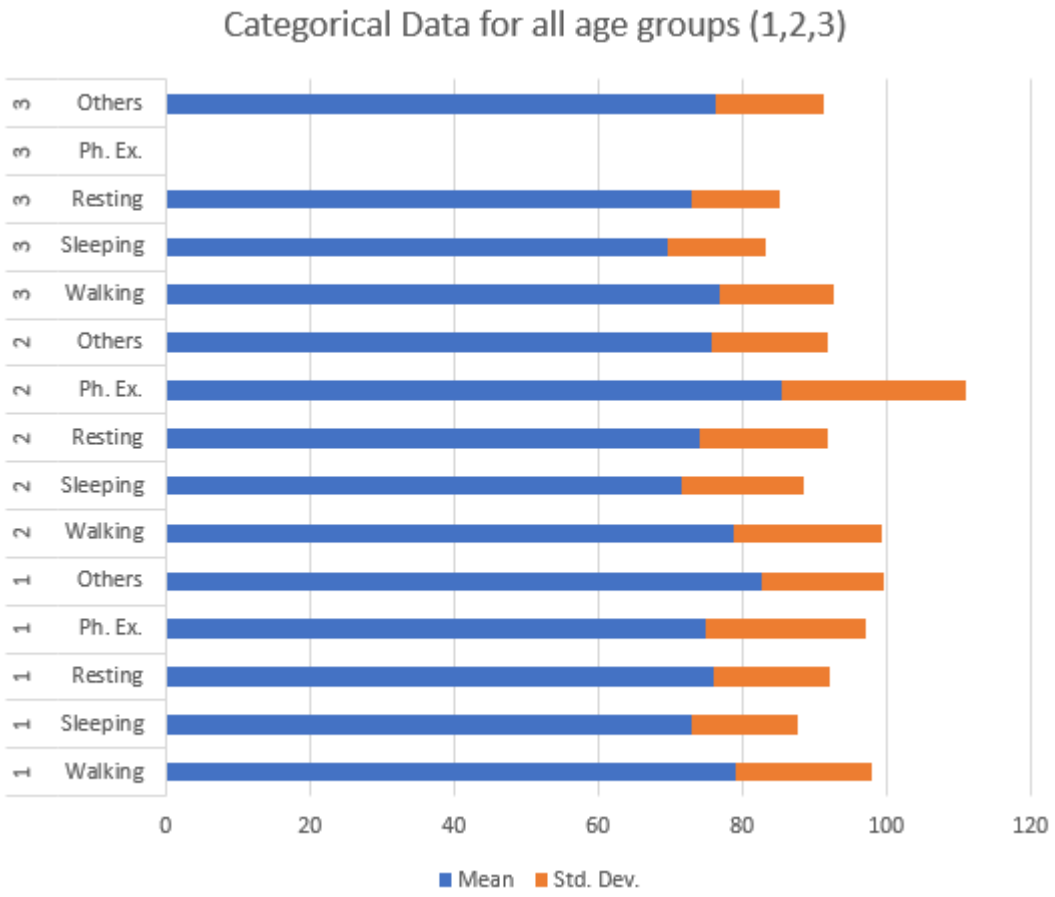


FIGURE 4.6: Visual Representation of Global Categorical Data.

4.1.4 Gender comparison

We also make a comparison between participants' genders, to see the different changes of data between male and female. This comparison can be observed in Table 4.3. Also, a comparison between genders inside each of the age groups was considered at the beginning, however this idea was discarded, as each age group had a very predominant gender. For example, 100% of test users in age group 3 were female, 83.3% of users in age group 2 were male; and 72.73% of users in age group 1 were male. Therefore, making this age-category-gender study would have been futile, as it would have been influenced by only one gender and the data from the non-predominant gender group would not be reliable.

For the gender study, the following information has been extracted, regardless of the age groups of all test users:

We can see the visual representation of Table 4.3 in Figure 4.7.

Note that, out of the 10 females involved in the study, **none** of them did perform physical exercise. Therefore, as physical activity was going to be compared between 14 males and no females, no conclusions can be extracted from making a gender-based comparison of *physical exercise* category data.

4.2 Conclusions

After concluding this offline, statistical study, several conclusions can be extracted:

Gender	Category	Mean	Standard Deviation
Male	Walking	78.07	18.71
Male	Sleeping	71.36	17.06
Male	Physical Exercise	78.33	23.11
Male	Resting	76.12	17.74
Male	Others	79.70	15.00
Female	Walking	78.85	18.23
Female	Sleeping	72.50	13.11
Female	Physical Exercise	??	??
Female	Resting	74.54	13.46
Female	Others	80.71	17.81

TABLE 4.3: Gender-Based Categorical Data

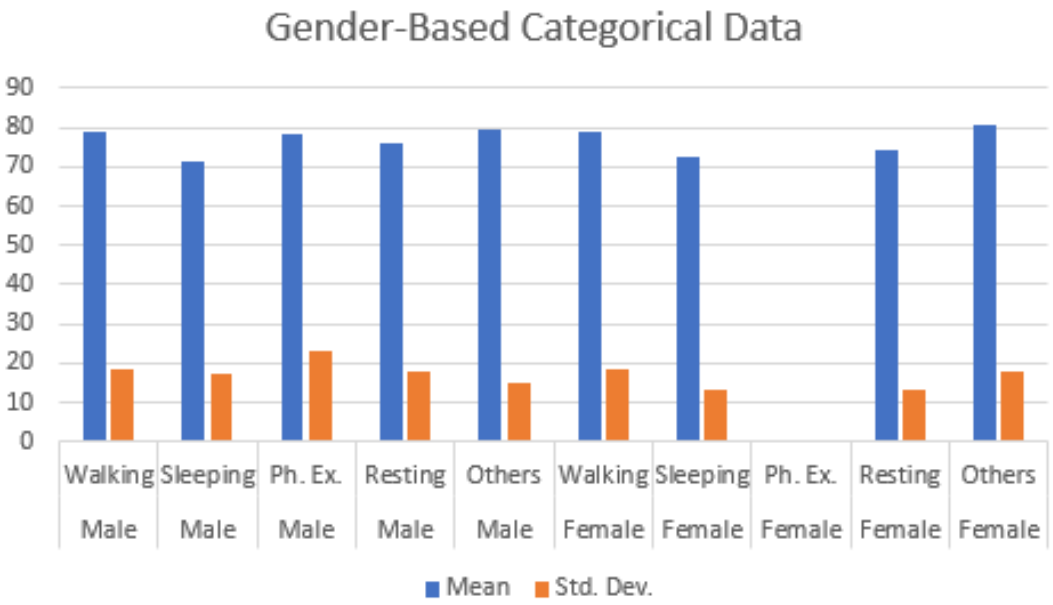


FIGURE 4.7: Visual Representation of Gender-Based Categorical Data.

- The sample size is insufficient, although statistical descriptors look robust. Even though the initial presumptions and predictions about the values that each age group or gender would have were correct, a greater sample size would have helped in order to validate with more confidence all conclusions extracted.

- The rate of frequency of measurements should have been smaller. However, this was already given by the GadgetBridge application's configuration options, as previously mentioned in Section 3.7, where it is assumed in all calculations that the heart rate measurement interval is 1 minute. Due to the GadgetBridge's limitations, this interval can't be lowered. Therefore, for this study, it was all that could be achieved. The normal proprietary software application from Xiaomi, Mi Fit, offers an interval of 2 minutes or higher, so this interval was actually halved. It is expected that, the real-time application, will have an interval of 6 measurements or more per minute, which will be six or more times faster than the offline statistical study.

- Users can often miscategorize activities, which can lead to invalid conclusions.
- During this study, it was suspected that several test users did not understand the difference between *walking* and *resting*. This is often caused by the daily routines of test users. Some people will interpret standing around as resting, for example, test users whose job forces them to stand up for several hours a day; and some other people will only consider resting as laying on a couch.
- The standard deviations observed, for all categories, age groups and genders, is too high. This is an inconvenience as false positives will be the norm, as mentioned in Section 6.1. However, the price value and the availability to access data is too big an advantage to ignore, in case of the Xiaomi Mi Band 2. Even though other wearable devices offer a higher measurement rate, it is at the expense of hardware cost. Also, note that several other devices, such as the Apple Watch 2, do not allow for the extraction of real-time heart rate measurements.

Having had a sample size of 24 users or more in each one of the age groups and gender groups might have been ideal for comparing this data in more detail, and extracting more conclusive information. However, doing this would have made the statistical study six times longer (or more) than originally predicted, because three different age groups and two different gender groups were identified. Including more users in each of these categories would have increased the time of the study significantly.

Chapter 5

Integration in a Social Robot

In this chapter, we will explain the development and integration of several modules with the social robot Mini Maggie. This robot is owned by the Department of Automation and Robotics, of the UC3M. The robot consists of hardware components, an Operating System (OS), ROS (see Section 2.3.2) and some software installed on top of it. The integration with Mini Maggie consists of developing software that complies with the standards of the Department. This software must be able to communicate with the rest of components that constitute the software layer of the robot. Regarding the initial data that is included into the software, we will use parameterized data from a user, so that the robot is able to distinguish users. For that, each user, from the robot's point of view, will be identified by his/her age and gender, and this information will be passed to the robot each time the software module is executed. We will detail the main architecture used for communication in the software layer, as well as the implementation of this software module, in detail.

5.1 Communications Architecture

In this section, the main application architecture will be explained. Considering that there are several design decisions that can be taken, there is a need to know why the producer-consumer was chosen amongst all of them. As explained in the introduction to this chapter, Mini Maggie uses ROS as a communications layer between software components. Therefore, it makes sense that the communications architecture used in the application resembles ROS's. The producer-consumer architecture is a design that allows communication between two entities. One of these entities develops data, and the other entity uses this information, therefore is the consumer. The communication between producer and consumer can be made through several techniques. Thanks to the utilities provided by ROS, the communication between nodes is made through the Internet: locally, if both nodes are in the same network, or through TCP/IP otherwise. As both producer and consumer nodes are located in the same computer, as there is no reason to execute them remotely on separate systems, this communication will then be made locally.
Through ROS topics, which are wrappers of a communication system that resembled the publisher / subscriber or producer / consumer architecture this communication will be achieved. The producer node will send a message by publishing it into a specified topic. The topic will be identified unequivocally by a name, and the consumer will subscribe to this topic, to asynchronously receive the messages sent by the producer. For a single topic, it is possible to have multiple producers and consumers, and there should not be a problem until messages are sent when multiple producers send messages simultaneously to a topic with a queue size smaller than the maximum number of producers. The existence of the subscriber is hidden to the

publisher and vice versa, as it is the purpose of this architecture to work independently, decoupling data production from its consumption.
Apart from the communication through messages via a topic, there is also the possibility to communicate through services, however this communication paradigm will not be used, as it is often more useful in distributed systems and architectures where n nodes wish to communicate with n nodes at the same time (n to n entity relationship). Since this paradigm is not necessary and only one node will be communicating with another (1 to 1 relationship), the producer-consumer architecture is the chosen one. In Figure 5.1 we observe an example of simplest publisher-subscriber system, with a 1 to 1 relationship.

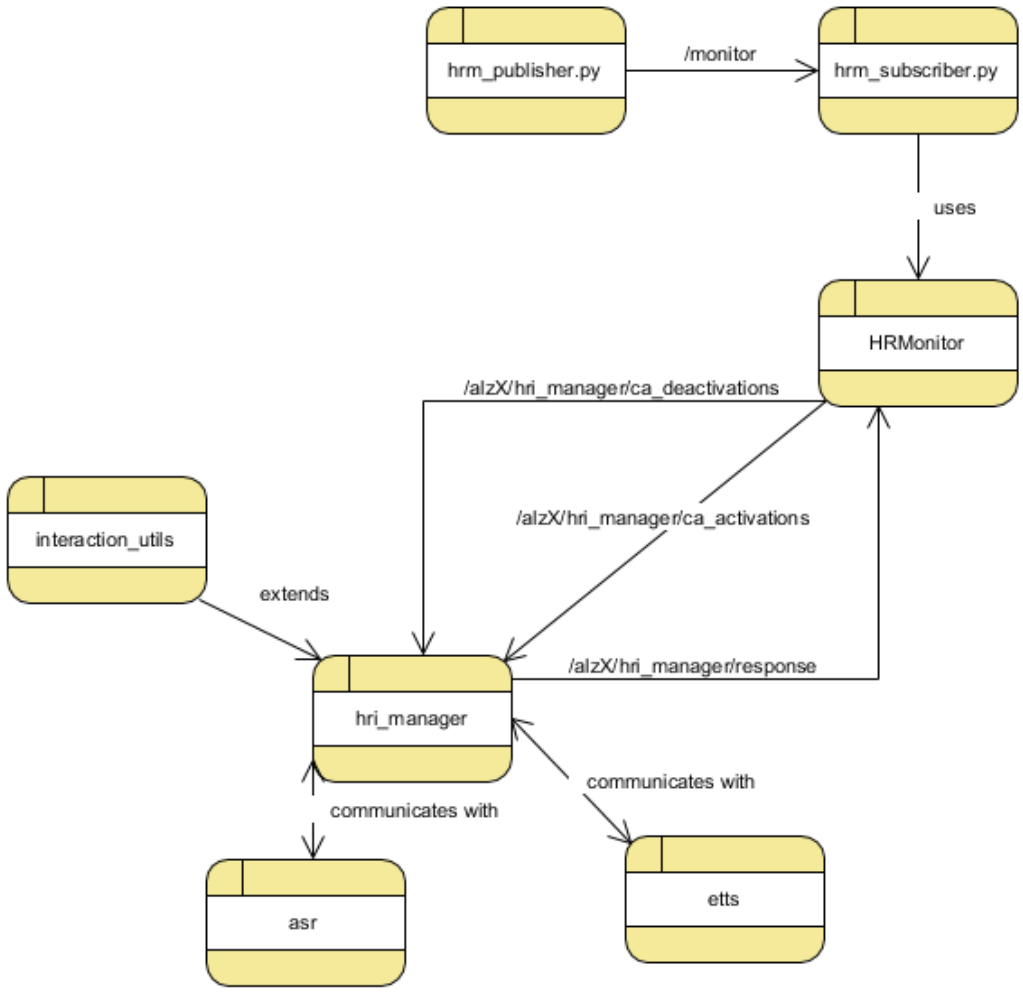


FIGURE 5.1: Publisher-Subscriber System and their interactions between them through ROS topics.

5.2 Social Robot Implementation

In this section, the main functionality of the social robot will be discussed, as well as the initial features and presuppositions in which the code is run. First of all, it must be noted that the application explained in this section, known as the real-time application. The main purpose of this application is to establish a Bluetooth

communication between a transmitter (Xiaomi Mi Band 2 device) and a receiver (the social robot). The duty of the social robot is the following:

- Periodically receive data from the Xiaomi band from a ROS topic, which correspond to the transmitter's detected heart rate measurement. This measurement will be asynchronously received, internally calling software interruptions to notify the the system of a new heart rate measurement happening. When this happens, a specific function callback will be executed.
- The program will store a total of 10 measurements in a sliding window. In case this amount is going to be exceeded by the insertion of a new heart rate measurement, the system will automatically drop one element from the queue to allow for an insertion. Otherwise, the heart rate measurement is simply inserted into the window.
- An anomaly detection system calculates whether a dangerous pattern has happened in the window. In case this happens, Human-Robot Interaction (HRI) begins to happen. The social robot will ask the user, using Speech Recognition (SR) techniques, whether he/she is feeling fine. If the social robot receives no feedback from the user, communication with a caregiver or family will occur, either by Telegram, SMS or Email communications. More information about the implementation of these modules can be found in Sections 5.4.1, 5.4.2 and 5.4.3.
- If the user responds positively to the interaction initiated by the social robot, everything is restored to normal.

The social robot will make use of third-party APIs (such as Telegram) and modules (such as university-developed modules, see Section 5.3) by importing their functionalities, and using them accordingly.

5.2.1 Data Structure

In this section, the main data structure used for the storage and handling of heart rate data received during the execution of the real-time program will be detailed. As explained before, it is a queue, based on a *dequeue* Python data structure, and its functionality is expanded with own-developed methods. This queue represents the time window, containing a maximum of 10 elements at a time. Depending on the rate at which the Xiaomi band retrieves information, which is usually less than every 10 seconds, this queue will fill up approximately in less than a minute. Moreover, the queue has been implemented so that it is configurable. This means that the size of the window can be expanded or reduced at will.
The expansion of functionality of the original *dequeue* Python object implements the following:

- Printing all elements in the queue, which is useful for debugging and displaying elements. It is also useful for building messages that can be sent to caregivers if necessary, in case of an emergency.
- Appending and deleting elements from the queue, which are basic queue operations
- Appending multiple elements into a queue at the same time, passing a list as argument to the function

- Reversing the queue and rotating it.
- Getting the size of the queue
- Calculating the average of all elements in the queue at any time. This is useful for threshold-based outlier detection in real-time. For example, if a user has exceeded its age and gender-based average continuously for the last 20 seconds, an alarm can be triggered to take proper actions.
- Deleting the queue.

5.2.2 Encryption Mechanisms

Xiaomi Mi Band 2's Bluetooth communications are originally *encrypted*. This is designed purposefully, so that private information about one's own wearable device is not accessible by everyone. Since encryption mechanisms are in place, information can not be retrieved trivially, and all communications that we want to do with the Xiaomi band must be done through encryption mechanisms. Once we have understood how the encryption of the Xiaomi band works, we will be able to communicate with it: send requests to the band (heart rate measurement requests) and extract the results from the request.

Encryption Algorithm

Internally, Xiaomi has programmed all Xiaomi Mi Band 2 devices to have standard AES encryption. AES is a variant of the Rijndael block cipher, being Rijndael a family of ciphers with a different key and block sizes. AES is a symmetric encryption algorithm, meaning that it uses an unique key both for the encryption and decryption of data. AES is currently unbreakable. It has some variants, the most common ones are AES-128, AES-192 and AES-256, in which the number in the name represents the key size. Typically, the block size is 128 bits. The key size specified for the AES cipher specifies the total number of rounds performed to the input. Typically, these are the number of rounds:

- 10 rounds for AES-128
- 12 rounds for AES-192
- 14 rounds for AES-256

The key size used by Xiaomi to encrypt data is 128 bits long. So, the current encryption algorithm is **AES-128**, which will transform, based on the encryption principles of confusion and diffusion, ten different rounds on the input to obtain the encrypted result.

Enabling communications with the wearable device

In this section, an overview of the encryption circumvention mechanisms used is be made. To allow communication with the wearable device, we make use of two auxiliar, publicly available applications: Wireshark and nRF Connect.

- Wireshark is a free, open-source packet analyzer. Wireshark will be installed in an Operating System to analyze packets from the mobile, produced when communicating via Bluetooth with the Xiaomi band.

- nRF Connect is a cross-platform tool that will allow us to scan and explore Bluetooth Low Energy (BLE) devices, and see their features, such as their GATT services. We will install this application in a mobile device to discover the Xiaomi band device and its characteristics, to subsequently know which characteristics to communicate with, in order to extract heart rate information.

Some of the encryption mechanisms used in this bachelor's work were already developed by Volodymyr Shymanskyy and Leo Soares ([32]). However, several modifications were made, as the initial libraries were insufficient for communication between the Xiaomi band and the robot. First, communication with the wearable device was not constant: we could only make one heart rate measurement at a time, and the Bluetooth connection was closed. Additionally, the code contained object-oriented programming errors that had to be fixed, and make it compliant to work with the rest of the code. Finally, as these initial libraries were supposed to transmit heart rate measurement information as a ROS publisher, ROS packages and libraries had to be imported and ROS code was developed from then on.

There are some concepts that need to be presented, in regards to the internal hardware of the Xiaomi device, as well as Bluetooth technology, before beginning to encrypt and decrypt data. Bluetooth Low Energy devices have some fields, that are present in any device with BLE technology. These are the fields:

- **GATT Services**: these services are a collection of characteristics of the device, and how it communicates with other services or devices. GATT stands for **G**eneric **Att**ribute Profile. It specifies the structure in which profile data is exchanged. So, this structure basically describes how to join together, transfer and present data to other devices using Bluetooth Low Energy. These services are a collection of data and associated behaviors that serve a purpose or characteristic. The definition of a characteristic is a value that is taken in a service, together with properties and configuration information, on how the value of this characteristic is accessed and represented.
  For example, a heart rate measurement service would offer a compulsory heart rate measurement value as characteristic. But, the wearable device has several services. For example, a battery level service, in order to observe which is the remaining battery percentage.

- **Descriptors** (e.g. read-only): we will have one descriptor for each one of the characteristics. For example, each characteristic has a descriptor that specifies if this characteristic is read only, or is used for notifications too. Some characteristics only have read/write permissions. In Xiaomi Mi Band 2, examples of this are the battery level or the current date. They, therefore, do not interact with the notification service.

- **Characteristics** (e.g. heart rate characteristic): there are some more complex than others. The more complex characteristics have a request/notification paradigm. This is the type of characteristic we seek: have a real-time heart rate measurement. For that, we will send several requests, and each one will return a heart rate measurement value.

- Note that every characteristic, service and descriptor is identified by a UUID. UUID stands for *Universal Unique ID*.

An example of a service, its associated descriptors and characteristics can be observed in here.

1. We use the nRF Connect application and seek a device. We find our Xiaomi band with the corresponding MAC address.

2. We find a GATT service. For example, the one corresponding with the Authentication service: *0xfee1*. We can see this in Figure 5.2.

3. We find the corresponding authentication characteristic, inside the GATT service, because there are several characteristics for each service. Some descriptors have a name, and others have an *unknown characteristic* value. To determine which one is the one we need, we found this out through Leo Soares' research. It is identified by the UUID *0000fec1-0000- 3512-2118- 0009af100700*. We can see this in Figure 5.3.

As explained before, for debugging purposes on the list of services, their corresponding characteristics, and the optional descriptors for each one of the characteristic, an Android application is used, called **nRF Connect for Mobile**. We can observe all this aforementioned information in Figures 5.4, 5.2 and 5.3. This application is one of the many publicly-available BLE debugging applications.

When connecting to one of the Xiaomi Mi Band 2 devices, a list of client services and their UUIDs are shown as seen in Figure 5.2.

As all UUIDs of the list of available services in the Xiaomi devices have been obtained, they can be defined as constants in our program. When communicating with these services, byte-level code will need to be sent, which will allow communication with each of these services, either for requesting data (if read descriptors are enabled for a specific characteristic) or for creating notifications (in case the notify descriptor is enabled). By looking at a Wireshark capture, captured in an Android device, when performing a Bluetooth pairing with a device, we can observe the packet trace in Figure 5.5.

As observable in packet number 685, the Bluetooth Attribute Protocol establishes that the service responsible for the **authentication** service is identified by the handle *0xfee1*. Referring back to the Android BLE debugger, we can see that this service in fact exists, with several characteristics inside:

From the Android application, the UUID of each one of the obtainable characteristics by this service can be obtained. Since this service has been already identified as the *authentication service*, this information can be observed as well in Figure 5.5. There, we can observe the handle for the main service, *0xfee1*. Then, we can cross-reference with this information with data from Figures 5.2 and 5.3.

- The handle for the main service is *0xfee1*
- The main service UUID, associated to the main service, and found by the Bluetooth BLE debugger Android application, is *0000fee1-0000-1000-8000-00805f9b34fb*
- The authentication characteristic taking place in the packet transmission has the following UUID: *00000009-0000-3512-2118-0009af100700*
- The notification descriptor handle is *0x2902*

With this information, we can proceed to the next section, where we will use all this obtained information to pair the device using an encrypted handshake mechanism.

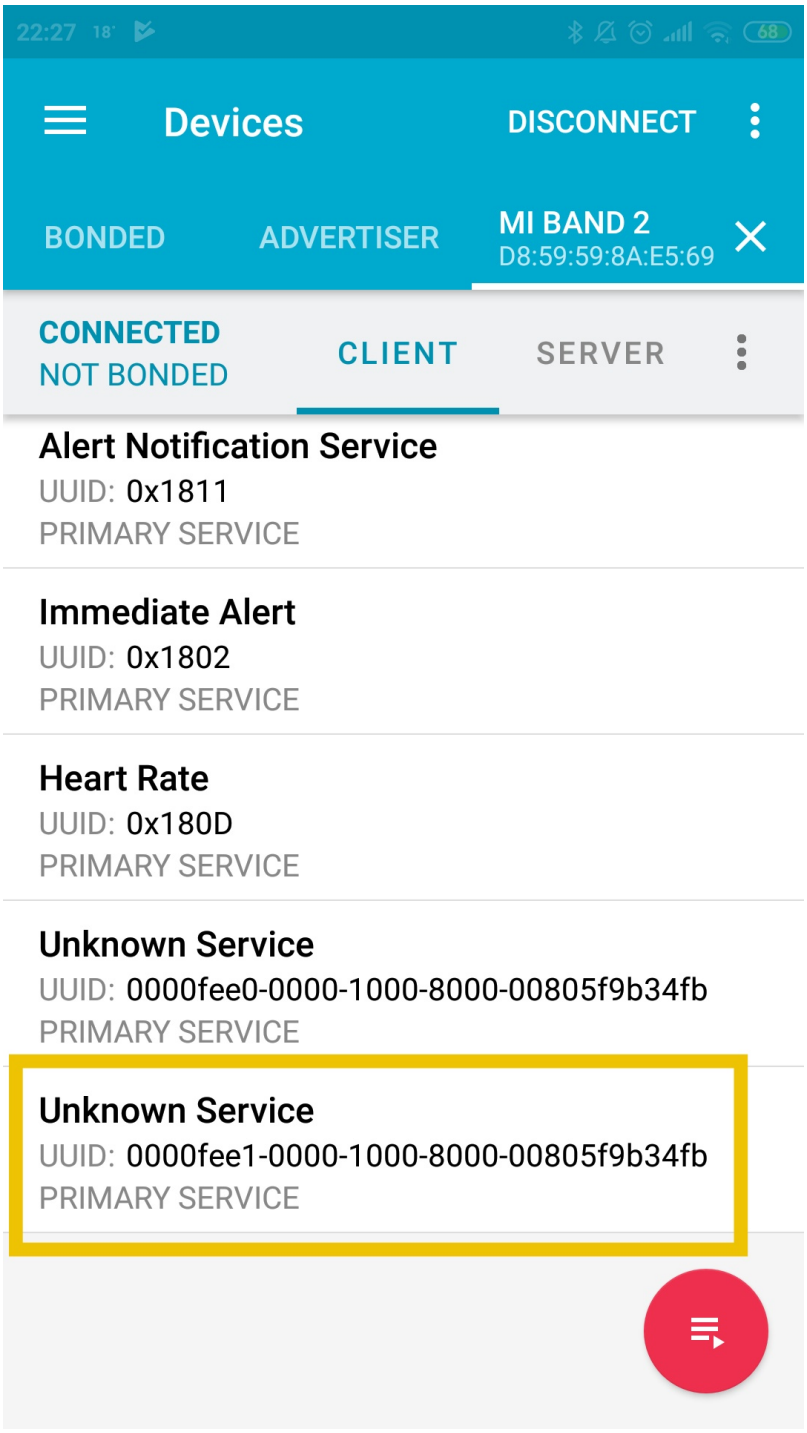


FIGURE 5.2: We can observe the authentication service.

Pairing with Device: Encryption

With all this information, and analyzing the Wireshark packet trace in more detail, the authentication byte-level exchange performed between the Bluetooth BLE device and the Android device is inferred.

1. First, we allow the Xiaomi band to open up a communication path between the mobile device and itself. This is done by sending bytes to enable *authentication notifications*. These authentication notifications are enabled by sending 2 bytes

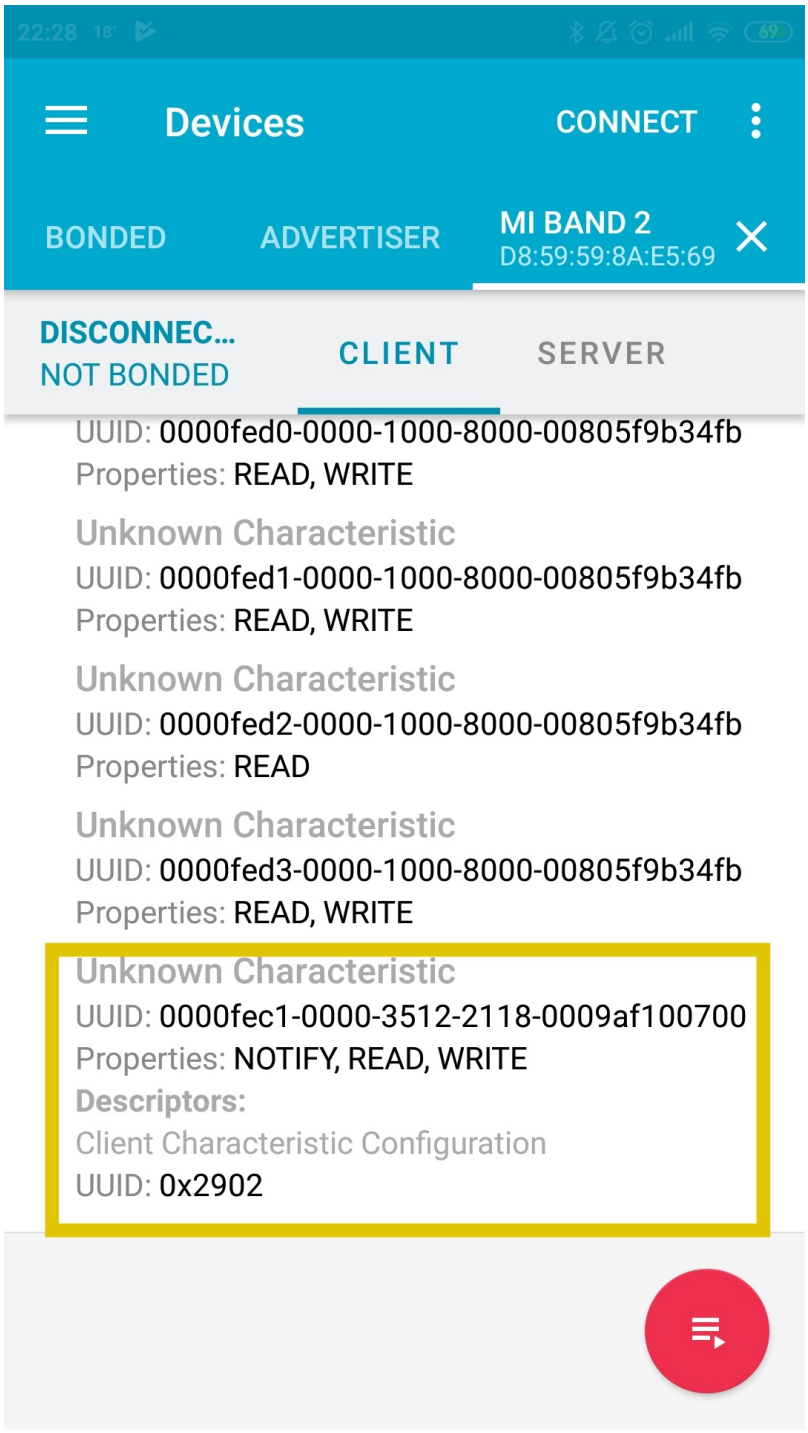


FIGURE 5.3: Here, we see a list of all the authentication characteristics. The bottom one corresponds to the one we want to communicate with.

to the notification handle (*0x2902*): *0x01* + *0x00*. (They can be disabled as well, by sending the following bytes to the notification handle: *0x00* + *0x00*).

2. Secondly, we send an encryption key, created by the mobile device, to the Xiaomi band. This encryption key will be used in the handshake exchange. We send a 16-byte long encryption key (remember, AES-128 protocol is used for encryption), meaning, 128-bit long encryption key. We send this key to the

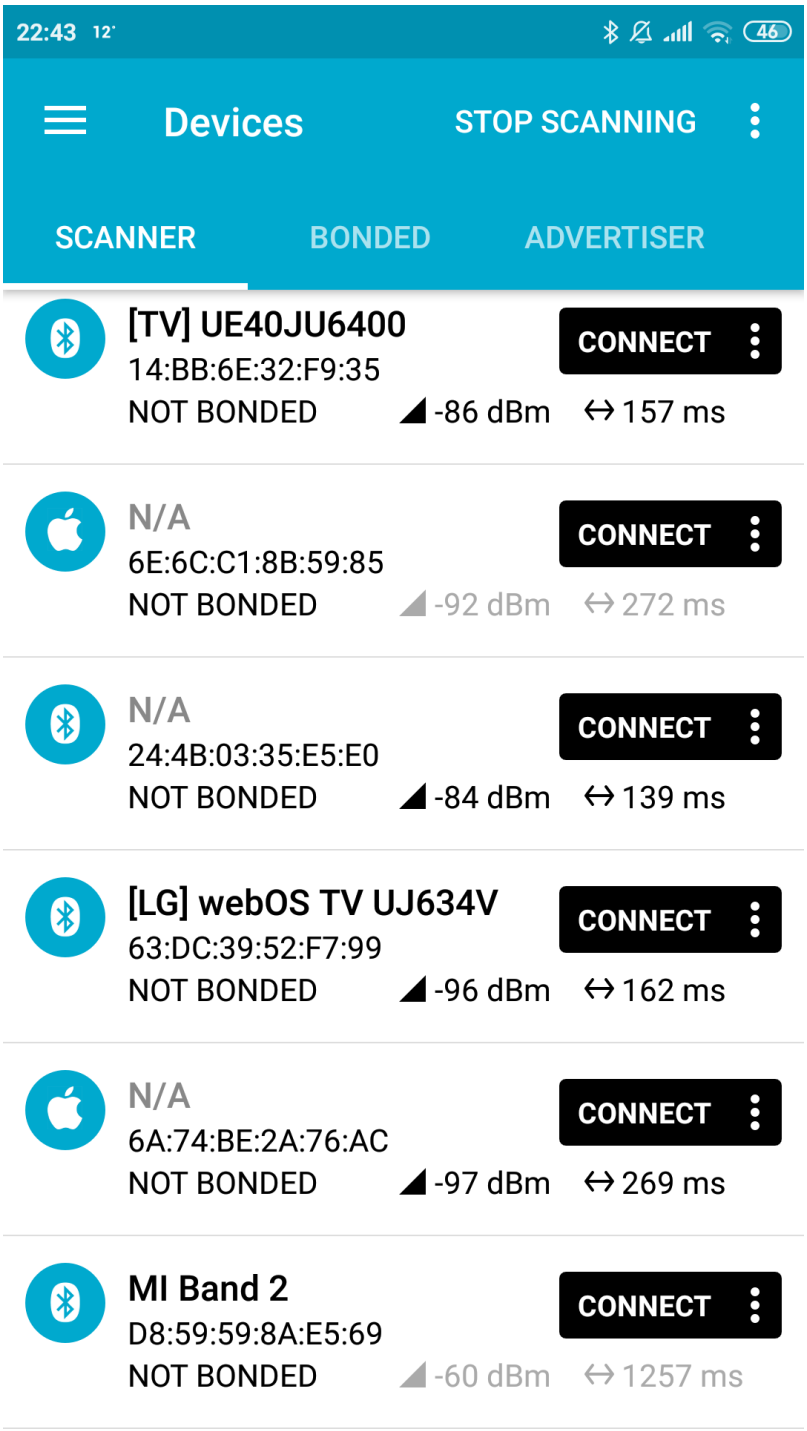


FIGURE 5.4: Nearby Bluetooth devices list using nRF Connect for Mobile

authentication characteristic appending it to the previously sent 2 bytes: *0x01 and 0x00*. Therefore, the final message has a format exactly like this: *0x01 + 0x08 + ENCRYPTION_KEY*

3. Third, we request a random number to the Xiaomi band, from the mobile device. As we perform a handshake, the Xiaomi will create a pseudo-random number and send it to us. To do this request, we send two bytes to the **authentication characteristic**: *0x02 + 0x08*. The random number is obtained as a

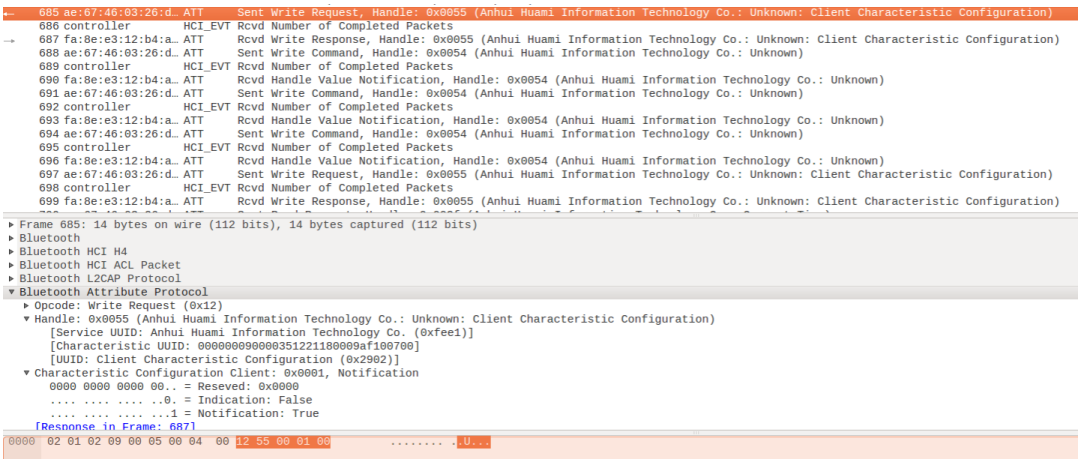


FIGURE 5.5: Wireshark packet trace of the encrypted authentication handshake mechanism, BLE

response (the last 16 bytes of the response).

4. Fourth: we encrypt the random number and send it back to the Xiaomi band. Once the Xiaomi band receives the number, it will decrypt it with the encryption key we sent in Step 2. If it matches, the handshake will have been successful. The random number from Step 3 is encrypted with the encryption key and sent back to the **authentication characteristic**. The encryption algorithm is AES-128, and is characterized by including **no padding** and **ECB** block cipher mode of operation (which is one of the many types AES-128 supports). The final message is: *0x03 + 0x08 + RANDOM_ENCRYPTED_NUMBER*.

5. Fifth: In order to comply with an encrypted authentication mechanism, as explained in Step 4, Xiaomi decrypts the random number with the encryption key and checks if it is the same value that it sent to the mobile device. In that case, both devices will be paired successfully.

After this, the authentication handshake between the Xiaomi BLE device and the Android device is finished. As this protocol is independent of the connecting device (it can be an Android device, the Bluetooth card of a car, a social robot...), this handshake process will be repeated exactly in the same manner when connecting to the social robot as a ROS node. A depiction of all exchanged data can be observed in Figure 5.6.

Reading data after handshake

Once the authentication process/handshake has finished between the social robot and the Xiaomi device, other services from the Xiaomi device become available. In these services, we can encounter information such as accelerometer and heart rate measurement characteristics, which is what is needed. Referring back to the Android application, a list of services and their respective UUIDs can be found through nRF Connect:

- Heart Rate Service, with handle 0x180D, and UUID *0000180d-0000–1000–8000–00805f9b34fb*
- Heart Rate Measurement characteristic, with handle 0x2A37, and UUID *00002a37–0000–1000–8000–00805f9b34fb*

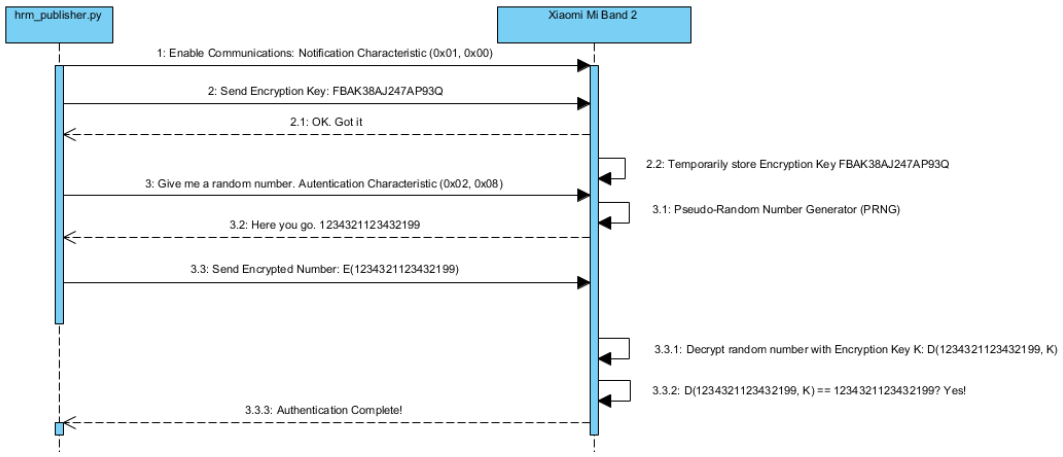


FIGURE 5.6: Data exchanged within the encrypted handshake, which is required to pair the Xiaomi band with the mobile device, or, in the end, the ROS node.

- Heart Rate Control Point characteristic, also called as the Heart Monitor Control Point Characteristic (HMC), with handle 0x2A39, and UUID *00002a39–0000–1000–8000–00805f9b34fb*

These are the three UUIDs needed in order to extract heart rate information, although the one we communicate with is the second one. The Android application, however, offers also additional UUIDs that are interesting, such as a sensor characteristic (SENS), from which accelerator values can be obtained; also, a notification descriptor, to send our own notifications to the Xiaomi BLE device.

All these procedures were translated into code as explained. However, there were some byte-level issues that needed to be resolved. For example, further investigation from researcher Leo Soares ([32]) proved that there was a byte-level difference between sending heart rate measurement requests:

- Sending a request to the Heart Monitor Control Characteristic (HMC) with the following bytes: *0x15, 0x02, 0x00* produces a result of a simple heart rate measurement.
- Sending a request to the HMC with the following bytes: *0x15, 0x01, 0x00* produces a continuous result.

However, after testing this byte-level exchange, the heart rate measurement characteristic (HRM) only returned one value at a time. Therefore, it was decided to implement, with an infinite loop, infinite requests to the HMC with a 1 second delay. Therefore, as soon as the HMC is be able to send data, it will. The amount of measurements received by the HMC varies every minute, but is accurate enough for having a real-time measurement system such as the one that is needed. Typically, because the wearable device can't process as many requests as we send it, the rate of measurement ranges between 6 to 10 measurements a minute.

5.2.3 Machine Learning: Dynamic Activity Detection

Considering all analyzed data in Chapter 4, there must be a way to allow the final real-time application to automatically categorize this data or classify it. There are two possible ways to achieve this:

- One method that could be implemented is to simply pass as arguments, when executing the final real-time program in a user, its age and gender, and from there, load the range values extracted from the statistical study. This is the static solution, which is valid and computationally-friendly for a computer, but is inflexible. This option is used in the development of our module to detect dangerous levels of heart rate of the user, previously basing these thresholds on the findings obtained from Chapter 4.

- The second option, which is the dynamic solution, is to use **Machine Learning** techniques or algorithms to allow the real-time program to automatically learn from this data, and automatically classify real-time data into one of the activity categories. This has the advantage of being more intelligent and independent on its own, however the disadvantage is having a more computationally-intensive program. This is what we will discuss in this section.

We will use Machine Learning (ML) techniques to attain this *dynamic prediction system*. An example of the final functionality would be:

- The ML classifier is already trained with a 25% of test data and a 75% of train data, from all the study.

- The user is interacting with Mini Maggie and is constantly exchanging information through Bluetooth with the robot.

- The robot, for each new measurement, passed this new information and the last three values of the user through the ML Classifier.

- We obtain a prediction of the supposed activity the user is performing, based on the last three heart rate measurements.

Data Preparation

In this subsection, we will explain how we prepared all data, from having independent files (each containing information from user) until having the final set of data, which is passed to the Machine Learning classifier. Our dataset uses the Python DataFrame structure and additionally, the *numpy* library for numeric operations.

The initial structure of data, to pass to the classifier, is the following:

HR1	HR2	HR3	Age	Gender	Activity	Basal
76	63	96	23	1	0	63

TABLE 5.1: Example of data sample for the Machine Learning Classifier. We can observe there are three heart rate measurements, the age of the user, his/her gender (male = 0, female = 1) and the basal heart rate of the user. Before preparing the final file, the basal heart rate is calculated as the mean heart rate of a user when he/she indicated *resting* activity in the Android application. Therefore, the accuracy of the basal heart rate is much better than getting the mean heart rate during all activities, as it resembles how it is calculated in medicine. Also, we can observe the Activity the user is performing. Depending the activity introduced in the Android application, a numeric value was given to the activity. (0 = walking. 1 = sleeping. 2 = physical exercise. 3 = others. 4 = resting). This is the **feature** that the ML classifier will attempt to predict, given the other six variables.

We had to convert non-numerical variables into numerical ones through a process called **one-hot encoding**. Since the machine learning algorithm we use, called **Random Forest**, used Search Trees internally, we need to have numerical values in order for the algorithm to develop a robust search tree with only numerical values.

After having this data, we need to consider something: the Random Forest algorithm does not know anything about our data. Therefore, it does not know, initially, which variable has more importance or is more decisive when making a prediction about the activity. We see in our example in Table 5.1 that we have heart rate values that are much higher than, for example, the gender (which is either 0 or 1). Therefore, we must *normalize* the heart rate values, which are predominant, before passing it to the classifier. For normalization, we have chosen a technique called **feature scaling**. There are several algorithms for feature scaling, but for the heart rate variables, we applied them the equation in the following equation:

$$x' = (x - avg(x))/(max(x) - min(x))$$

where x is an original value, x′ is the normalized value.

After normalizing values, the initial data from Table 5.1 is transformed into what can be seen in Table 5.2.

HR1	HR2	HR3	Age	Gender	Activity	Basal
0.184198	0.126506	0.039995	22	0	0	80

TABLE 5.2: Normalized Values after performing feature scaling. This is the data that is fed to the ML Classifier.

After normalizing this value with the corresponding values, we indicate the ML Classifier which features are used for the prediction, and which variable is the one we want predicted. As explained before, we use a train-test split method, with a test size of 25% of all samples, and the rest (75%) will be predicted using this initial 25%.

Once the data preparation process is finished, we created a Random Forest classifier with 100 estimators, and trained the model using our training sets. The execution of this training is made once every time Mini Maggie launches. Predictions are made in real time, after we receive a new heart rate measurement, by comparing it with the trained classifier.

Algorithm Accuracy

After doing some tests in real time, and according to the accuracy score of the Machine Learning classifier, the total accuracy of the algorithm is **50.163%**, with a Mean Absolute Error of 1.32 degrees. That means that, 50 out of every 100 times, the classifier will be able to predict the correct activity of the user with a 95% accuracy. This, for a Random Forest classifier, is very inaccurate. As we will explain in Section 5.5, this lack of accuracy is probably caused by root causes in the data collection process, as well as from the high standard deviation from Xiaomi Mi Band 2 devices, which skewed the final data set. Therefore, we can conclude that the machine learning classifier is not working well for this data set, and for future studies, a better data collection process shall be considered (and/or using different measuring devices, such as a higher-quality, lower-standard deviation wearable device).

5.3 External Modules

In the Universidad Carlos III de Madrid's social robotics laboratory, RoboticsLab, other modules were used as additional supporting elements to incorporate and ease the programming difficulties found during the development of the medical alert system. These modules are useful to interact with the rest of Mini Maggie's systems. All systems work cooperatively to offer a seamless Human-Robot Interaction.

We can find some of these modules, with a detailed explanation of their added functionality here. Also, a reason why they were used to communicate with our own module. Note that, apart from this explicitly-used modules, there are several more that are implicitly executed, whenever the full functionality of the social robot is launched. However, they do not communicate with our own module. As the system has too many parts, and none of those interfere with this bachelor's work, they will not be mentioned here:

- HRI Manager: it is in charge of providing a set of functions for handling packing and unpacking of data. This is necessary in connection to ETTS, as the result of a ASR message contains a dictionary with several information. All these values can be unpacked using a HRI Manager function, which converts a speech recognition result to a dictionary, and allows further inspection of this data in an easy way.
  The HRI Manager makes use of Communicative Acts (CAs): they are an abstraction that represent the interaction with the social robot. These acts define an action by the social robot. For example, if the social robot moves its right arm, a communicative act must be launched with this pre-defined, parameterized action.

- Text-To-Speech (TTS): its purpose is to give a voice to the robot, by uttering text, previously passed as input. The TTS module hides the complexity of the etts engine running behind it (Nuance, Loquendo...). When using this module, it is also possible to specify emotions, in order for the social robot to act worried, anxious, happy or mad. The etts module runs inside a Docker container.

- Automatic Speech Recognizer (ASR): it is a module used for the recognition of speech input from a user. Through grammars, the social robot will use ASR software to detect the speech of a user, and run it by the grammar. Afterwards, the engine will produce an output of what the user said. The ASR module also runs inside a 64-bit Docker container.

- Interaction Utilities *Utils*: this module allows the use of communicative acts in a transparent way. Through the use of this module, all types of communicative acts can be launched. These involve from the movement of a body part of the social robot, to asking a question to the user with ASR or through the interactive screen of a tablet.

- Interaction Messages *Msgs*: this module is used as it contains the definitions of standard message types. These message types involve the structure of a **Communicative Act** message, and the structure of a **Communicative Act Result** message. Using these structures, information is obtained from ROS topics in an established way, which can be analyzed and modified.

All used external modules are useful for interacting with the robot. All data analysis, data structure handling and operations are made in the background, with

no HRI at all. Therefore, it makes sense that our own module interacts with the robot in the only HRI component present in this bachelor's work, which is through the ASR module. A depiction of the communication between our own module and these aforementioned modules is found in Figure 5.1.

5.4 Own-Developed Modules

With the help of external APIs (Application Programming Interfaces), made available to the public, some additional modules were developed, to increase the functionality of the social robot, as well as to allow communication of the social robot with the outside world. This is especially necessary, as part of this work consists in implementing an alert system, which can aid a person in need, whenever the program detects a person is in trouble. For that, communication with emergency services or relatives of the patient is *essential*.

Let us have an overview *example* of a situation in which the social robot can interact with the own-developed APIs and how this communication could be useful. This is the final implementation that has been considered in our module.

1. Our module communicates through Bluetooth connection with the Xiaomi band, constantly making heart rate requests through corresponding GATT service. It stores it in the data structure explained in Section 5.2.1 and calculates the danger level of the last ten measurements.

2. In case the user's heart rate has been constantly in danger for approximately 20 seconds, alarms are raised..

3. The social robot interacts with the patient and asks if everything is okay.

4. If the user does not respond in 10 to 15 seconds, the social robot stores information into a file which can be read by the Telegram file. If the caregiver or assistant sends a request to the Telegram module, he/she will have access to a history of the heart rates of the patient, and an indicator saying whether the patient is in danger or not. Optionally, if it receives a message from a family member (included in a pre-defined list) asking for the robot to help the patient, Mini Maggie can make use of SMS, Telephone or Email modules to alert the corresponding parties in case of emergency.

5. The social robot's behavior will be constantly changing based on the responses received by family members or the patient's Telegram accounts. This behavior will be present synchronously in all own-developed modules, as they are additions to the main program's execution flow.

This constitutes the workflow of the final application. Now, we will explain the set of additional modules that were used to help the main functionality of the application, which are the Telegram bot, an Email bot, a SMS bot, and some other additional modules that were not used in the end but are ready to be.

5.4.1 Telegram Implementation

As social media and instant messaging applications are the basis for mobile internet communications nowadays, the implementation and creation of an API for sending, receiving and responding to messages in Telegram was deemed necessary. As WhatsApp, which is currently the most widely-used instant messaging application

in Europe, does not supply any facility or API for communicating or creating bots, the creation of a WhatsApp API or bot was discarded. However, Section 5.4.4 discusses the possibility of renting a service for WhatsApp message handling. Telegram was, therefore, chosen for its many facilities given to developers.
For the development of this API, a Python 3 module called **Telethon** was used. Telethon is not another Telegram API implementation, it is a wrapper of the original Telegram Python module, but removing much of the complexity from the original module. Telethon allows communicating, based on a Token that can be obtained from the official Telegram developer website, with any other bot or person identified by a user name in Telegram. Information for each of the *bots* is sent in an encrypted session file.

Note that sending messages to a family member of a user in distress is the original purpose of implementing a Telegram API. Additionally, an asynchronous implementation for **responding** to messages of these family members has also been implemented.

5.4.2 E-mail Implementation

As an additional feature, a SMTP Email service has been developed, to allow communication with emergency services, or any other email recipient, in case the patient is in need. It was decided to implement an email service, because email is one of the main communication mechanisms, aside from instant messages and social media, that is used nowadays to communicate between each other. It is a free service, which shall be considered as an advantage, as not every own-developed module is free.
SMTP stands for Simple Mail Transfer Protocol. It makes use of a DNS server internally, in order to know its own SMTP initial server. The SMTP email service contains a DNS server. This DNS server allows for several records. There is a specific DNS record, called the MX (Mail Exchanger) record, that allows to translate (from an email address, its corresponding IP address) and specify the mail server that will be responsible for sending messages in the name of the user.
The official SMTP Python API automates this DNS record-calling process, as well as several other calls necessary for sending messages to a recipient. All DNS queries will be made by the own-developed module, to *smtp.gmail.com*, to port 587 (which requires a TLS connection).
The initial authentication mechanism requires a personal email address and its corresponding password. As a security measure, obtaining this information is made through the use of environment variables, in which the user can establish some variables in the computer, so that no personal confidential information, such as the email plus password are stored in the original application code.
The SMTP library allows for a wide variety of calls, so a smaller API was developed, in order to collect the most essential kinds of messages and calls that will be made to the SMTP server. As the main purpose of this service is to send simple messages, without attachments, a simpler implementation is sufficient to satisfy the application's needs. So, sending a basic email, with a text, a target address, and a sender's address is enough. However, the smaller own-developed API allows for sending complex emails, with encoded attachments.

5.4.3 SMS Implementation

Through the use of a company that provides all kinds of APIs (for voice communication, SMS communication, programmable chatting...), called **Twilio**, a SMS library has been developed. Twilio offers several mobile phones, with voice calls and/or SMS capabilities, that can be rented monthly. The cost is as low as 1 dollar/month. Therefore, a SMS implementation was thought to be cost effective, as the pros of using this service seem higher than the cons.
For the development of this module, several SMS functions, taken from the official documentation from Twilio, were implemented. The set of capabilities that the module currently supports is:

- Sending a text SMS through the API provided by Twilio
- Monitor status of already-sent messages, by implementing callback URLs.
- Sending MMS, which are multimedia messages. Any *.gif*, *.png* and *.jpeg* files are supported.
- Obtaining a list of all messages exchanged with a specific phone number, or a subset of this list of messages.

The API from Twilio supports many more calls, as well as a RESTful API for the messaging service. However, as no web service is provided in this case, the RESTful API was left out of the implementation for now.
Note that credentials from Twilio are obtained from environment variables, in the same way the SMTP Email implementation does in Section 5.4.2.

5.4.4 Additional Possible Implementation: Mobile Calls + WhatsApp

Additionally, in the Twilio module, some calls were implemented and were left untested, since the cost for renting a Twilio mobile phone with a voice call capabilities was much more expensive than a mobile phone with SMS capabilities. The call and response mechanisms are implemented in the SMS module as annexes, and they allow to handle automatic calls to the social robot.
Finally, Twilio offers a new service that allows sending WhatApp messages. This implementation was also made, but left untested due to budget capabilities.

5.5 Summary

In this chapter, we have seen how our application is integrated into the social robot Mini Maggie. Also, we observed the technical difficulties that happened when we tried to see how the Xiaomi band internally and externally communicated data, and the lack of information of structure from this data. After this study, the encryption process, from handshake to data exchange, is completely clear, which will give us a clear view of the Xiaomi system for the future publication of a scientific paper.

With regards to the implementation of a Machine Learning classifier to predict activities, we have learned that the predictions with the present data are either inaccurate or not very good, due to the faulty accuracy of measurements. This can be caused by the high standard deviation of measurements, as explained in Section 6.1, or having the human factor take part in the data collection process. As users had to indicate their activities, if they forgot, or introduced an activity 10 or 15 minutes

after initiating it, a percentage of all this data would have been miscategorized. We believe this was a mixture of both factors: the high standard deviation of Xiaomi Mi Band 2 and the human factor.

Finally, with regards to the final implementation within Mini Maggie, the application has complied with all the standards required to properly communicate with other modules, and the application works successfully.

Chapter 6

Tests

In this chapter, various tests will be explained and shown in regards to:

- The Xiaomi band, used in Chapter 4.
- The Android application explained in Chapter, 3.
- The final robotic application, explained in Chapter 5.

6.1 Studying Xiaomi Mi Band 2's reliability

In this section, a study of the reliability of the measures provided by the Xiaomi Mi Band 2 devices will be made. As these are the devices that have been previously studied for all the data that is going to be analyzed in Chapter 4, choosing the proper device is an important decision. Ultimately, these will be the hardware devices that will implement the final product, that will allow the interaction and communication with the social robot in real time.

Having studied previously what is cited in [33], the heart rate measures provided by Xiaomi Mi Band 2's are, amongst its competitors in the era - conceived by the manufacturing date of Xiaomi Mi Band 2, which is in 2016, comparing with the following hardware devices that were also relevant in this year:

- Apple Watch 2
- Samsung Gear S3
- Jawbone Up3
- Fitbit Surge
- Huawei Talk Band 3

It has been proven that the variation in the measurement for the user's heart rate was the lowest amongst all the studied hardware devices. This following table ([33], Table 4) reflects the accuracy and stability of the measurements by each of the hardware devices previously mentioned, also including two software applications that were installed in the mobile device of the user. It is concluded, by analyzing the data, that the Xiaomi Mi Band 2 performed relatively *poorly* compared to the rest of devices, having the highest standard deviation, compared to the Samsung Gear S3, Apple Watch 2, and Fitbit Surge:

Measures/Device	Mean(SD)	Minimum
Samsung Gear S3	0.04 (0.03)	0.00
Apple Watch 2	0.07 (0.08)	0.00
Fitbit Surge	0.08 (0.12)	0.00
Xiaomi Mi Band 2	0.12 (0.13)	0.00

TABLE 6.1: Heart Rate Measurement differences between wearable devices ([33], Table 4).

These results put the hardware used in a preliminary negative and pessimistic position, as this means that there will be a higher variability in the heart rate's measurement of a user, causing higher and lower heart rate measurements more frequently than other hardware devices would. However, the study does not contemplate the hardware defectiveness of the Xiaomi Mi Band 2 device, as the methods used included a total of 44 healthy subjects plus 6 hardware devices (those mentioned before in this section). However, as my study is based on three different instances of the Xiaomi Mi Band 2 product, the reliability between different Xiaomi Mi Band 2 devices must be studied (in this case, three different devices). This is not contemplated at all in [33].

Therefore, the purpose of this section is to study the reliability between the three different Xiaomi devices, pre-assuming that the measurements won't be ideal as if comparing or performing the same study with three other devices, such as the Apple Watch 2, as explained before.

Heart Rate Variability Study

To perform this study, three test users were selected from the already-existing set of test users of the quantitative study. This quantitative study will be made in Chapter 3. These users were asked to perform two different tasks repeatedly:

- Whilst wearing the devices, one in each wrist, and the third one in a wrist randomly chosen, submerge both hands under water while a measurement is being made, to test the reliability under water. This procedure was repeated 20 times for each test user.
- Whilst wearing the devices, one in each wrist, and the third one in a wrist randomly chosen, jerk their arms around, in an attempt to make as much movement as possible and confuse the sensors, while a measurement is being made. This procedure was repeated 20 times for each test user, as well.

As there are three different devices, in order to differentiate them, their Bluetooth MAC address was used, to distinguish the data coming from each device separately. The three different MAC addresses - extracted from the device as these are the addresses used to interconnect the Xiaomi device to the social robot via Bluetooth technology -, are:

- ED:61:38:85:D1:FC, which will be called *Device 1* from now on.
- D8:59:59:8A:E5:69, which will be called *Device 2* from now on.
- E6:A1:BD:94:31:54, which will be called *Device 3* from now on.

The results of the study are shown below in Table 6.2. Emphasis is made on the inter-measure standard deviation, as it is an indicative of the accuracy between same measurements in different wearable devices. For example, if a user has made one measurement, and each wearable device returns: 50, 70, 90, the average of this measurement is 70, with a standard deviation of 20. Therefore, the difference between measurements of all three wearable devices is high. However, in reality, the standard deviation observed is much smaller, which indicates that all three wearable devices obtain very similar values when doing simultaneous measurements on a test user, even in disadvantageous conditions such as submerging all three devices under water.

User	Total Measurements Average (bpm)	Standard Deviation
1	67.98	3.75
2	80.49	3.83
3	67.63	3.24

TABLE 6.2: Hardware Reliability Study Data

As observable, the standard deviation between all measurements of users does not exceed 4. This means that, calculating a global average of all measurements, typically all values are within +-4 values of this average value.

This concludes that the study will be, at least, consistent between test users, which allows the comparison of data between Xiaomi Mi Band 2 devices. Had the data not been consistent and reliable within the Xiaomi devices themselves, data analysis would not have had any purpose or made any sense.

As it has been shown before that the reliability of the measurements will be worse than if they were made with other hardware devices ([33]), the complexity of this problem will rely on the effectiveness of the algorithm and the differentiation between false positives and actual data from the user. The interaction with the social robot will also greatly reduce the impact of the inaccuracy of some of these measurements, as HRI (Human-Robot Interaction) will ask the user, when in doubt, of any potential on-going risk or harm the user might be suffering. It must be finally noted that the reason for choosing a Xiaomi Mi Band 2 device instead of other devices, such as the Apple Watch 2, Samsung Gear S3 - which both have a smaller standard deviation in the study previously mentioned - is the price. Mind that one of the main purposes of this bachelor's work is the development of an *inexpensive* system. Systems with a smaller standard deviation have higher current price, in 2019, of 10 to 15 times the price of a Xiaomi Mi Band 2 device.

6.2 Android application

A graphical representation of the application can be observed, as a visual aid for this explanation, in Figure 6.1.

As mentioned in Figure 6.1, we have three separate activities. The user will be able to select one of the five distinct activities in the menu. Afterwards, the user will be prompted to introduce, in one of the two possible ways (a radio button or a progress bar) the duration of the activity, and confirm it. A visual demonstration of the application can be found in [1].

[1] https://youtu.be/910GbbF2sco

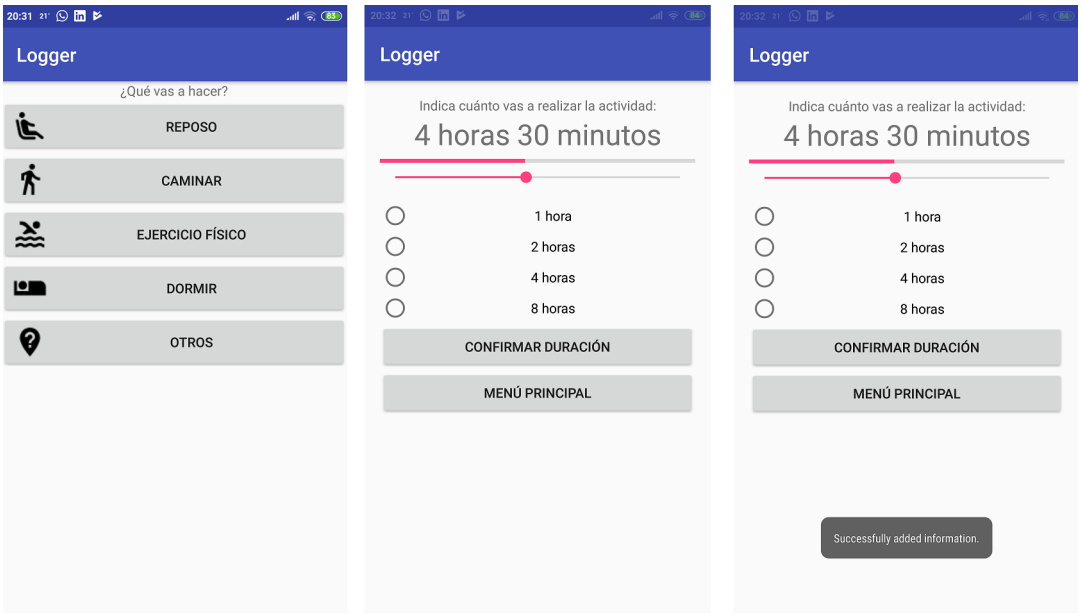


FIGURE 6.1: In this Figure, we can observe the homepage duration activities (Figure 1 and 2). In Image 1, we observe that the user is able to select one of the five different options for activities. in Image 2, the user selects the duration of the activity. Finally, when the user confirms the duration by clicking on the *Confirmar Duración* button, a toast message is displayed at the bottom of the screen, as we can observe in Image 3.

6.3 Robotic application

In this section, we will demonstrate the functionality of the final robotic application. For that, we propose three different use cases. Additionally, we will show how data flows between the application and what the robotic application really needs and does with the data it receives, in Section 6.3.1 (the first use case).

6.3.1 Use Case 1: user resting whose heart rate raises rapidly

This use case happens when the user of the application was, for example, resting and starts doing physical exercise. When the user puts on the wearable device, the robot will start requesting heart rate measurements and store them in the data structure. We can observe this in Figure 6.2.

Then, it will perform the average of the last ten measurements. Every minute, the average will be inserted in the ML predictor, in order to predict the activity the user is performing at that exact time. For a most accurate ML model, we will need 3 minutes from the beginning of the execution of Mini Maggie to have a correct predictor, as the ML predictor is initially filled with the basal frequency of the user.

In case the user's heart rate is higher than the average of his/her age group for physical exercise (as calculated and shown in Table 4.1.3), the *danger level* will increase by 5 percent. Once this danger level reaches 80%, Mini Maggie will ask the user if he/she is alright. Two scenarios happen now.

- Scenario 1: the user answers Mini Maggie, indicating that everything is alright. In this case, Mini Maggie reduces the danger level back to 0% as everything is okay, and continues measurements. This scenario is resilient to errors because

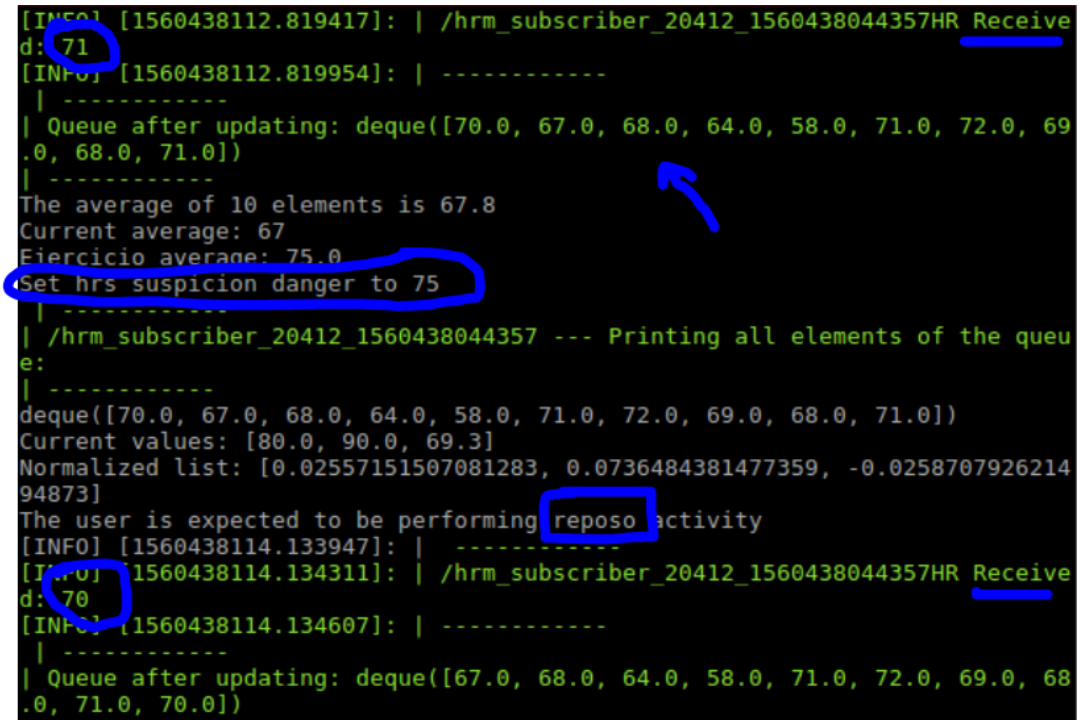


FIGURE 6.2: We can observe how Mini Maggie handles the data structure in two different iterations. We can observe how we receive data from the Xiaomi band, as well as the data structure and its contents. Additionally, we can observe the danger level it is currently in, and the predicted activity the ML model has predicted the user to be in.

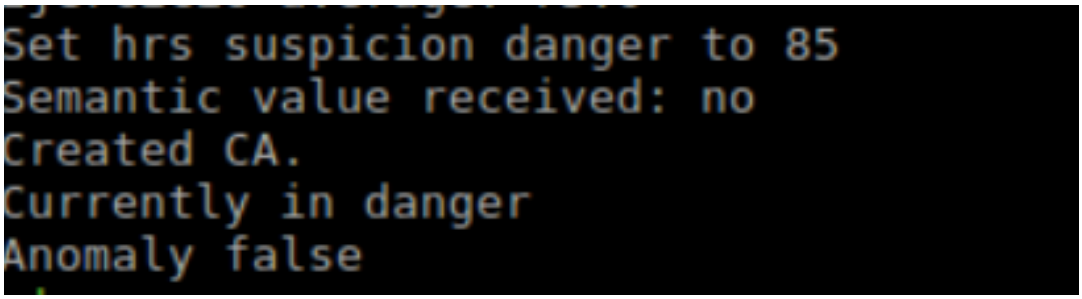


FIGURE 6.3: This is the data flow when HRI happens. We can observe that Mini makes a request through a CA, and waits for a response. Through the use of grammars and the ASR module, as explained in Chapter 5, the result is interpreted and a semantic value is received, which is the final result that we are interested in. As the semantic value received is *no*, the anomaly is classified as false and the danger value is reset back to 0.

it requires user interaction in case danger level is detected. Therefore, the user is always in control of the final decisions made by Mini Maggie.

- Scenario 2: the user does not answer. This scenario is contemplated in Section 6.3.2.

The data flow from Mini Maggie when this HRI occurs is shown in Figure 6.3. We can observe this use case in practice in [2].

[2] https://youtu.be/z7m5R6ornhI

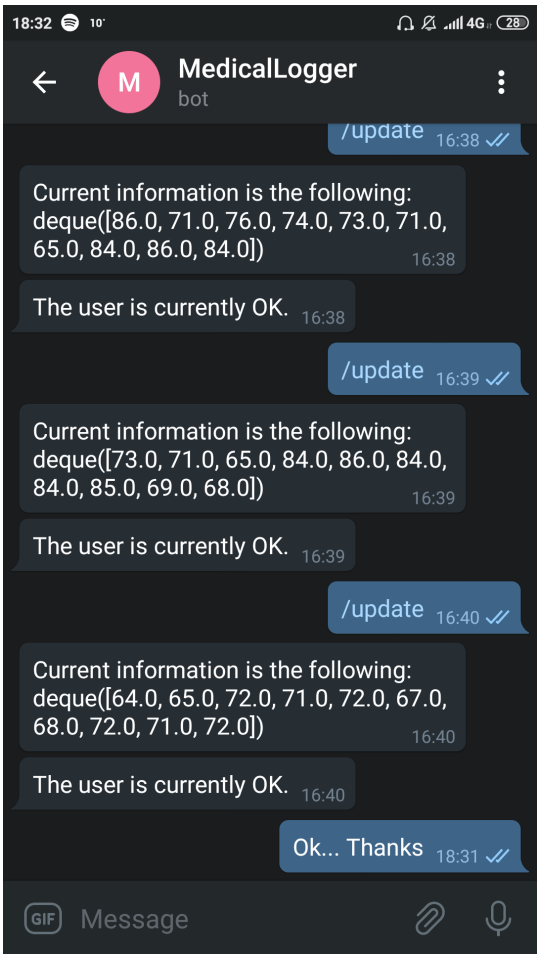


FIGURE 6.4: Telegram communication with the Medical Logger bot.

6.3.2 Use Case 2: user who has had a heart attack

This use case happens very similarly to the one in Section 6.3.1. The heart rate of a person that has had a heart attack will raise very quickly compared to its previous heart rate. We know there is no way to determine if the person has had a heart attack or has started doing physical exercise, however, we need to consider these two factors:

- The robot's final location would be in a user's household
- Doing physical exercise at home is less probable than doing other activities
- The *target audience* for the robot is people of old age that live in residences or hospitals. These people are much less likely to perform physical exercise at an advanced stage of their lives.

Therefore, we make a demonstration as well as the user having a heart attack, and **not** responding in the allowed time to Mini Maggie (20 seconds). In this case, the behavior will be different than the one in Section 6.3.1: now, the robot will initiate an internal procedure indicating that no response has been received, and this information will be accessible by requests through the Telegram bot. In Figure 6.4 we can observe a user requesting information from the user via the Telegram bot.

We can see an example of a user simulating to have a heart attack, and see how the robot would react when no response is received, in [3].

6.3.3 Use Case 3: user who has been resting for too long

In this last use case, we explain the case where a user has been resting for too long. This use case is more related to the ML part of the bachelor's work, as it will take the predicted activity of the user previously calculating it, by running it through the ML model.

In [4], we observe the case where the user has been sitting for too long in a chair. Then, Mini Maggie initiates conversation by recommending the user he/she should take a break from resting and activate their body. As the preciseness of the calculations made by the ML model are not so accurate, as explained in Section 5.2.3, this predictor needs to be improved in future work.

[3] https://youtu.be/w3X9r68JApA
[4] https://youtu.be/CubHnJ3Nuh0

Chapter 7

Problem Description

In this chapter, we will explain the set of requirements to develop the robotic application previously explained in Chapter 5. In Section 7.1 we explain the format that we follow for the requirments. In Section 7.1 we will explain the set of functional and non-functional requirements that need to be followed; and finally, in Section 7.2 we will make an analysis of all requirements.

7.1 Requirements

Requirements are a set of capabilities that a system must fulfill in order to satisfy a specification between a developer and the client. In our case, where we have no client, we will play both roles of the developer and the client. These requirements have a set of characteristics that are almost always common, regardless of the standard requirements must fill:

- Complete: the requirement must not be composed of several sub-requirements. It must fully state the requirement in itself, with no missing information.
- Traceable: the requirement must meet all parts previously stated by the client
- Verifiable: the completion of a requirement must be able to be proven.
- Unambiguous: the requirement shall not contain technical jargon or acronyms, and shall not express subjective opinions.
- Consistent: requirements shall not contradict one another.

The format followed to state the set of requirements can be seen in Table 7.1.

ID	**X**
Necessity	**Necessary / Non-necessary**
Priority	**High / Medium / Low**
Stability	**Stable / Not Stable**
Verifiability	**Verifiable / Non-verifiable**
Status	**Proposed / Verified / Validated / Rejected**
Type	**Functional / Non-functional**
Name	**Name of the requirement**
Description	**Description of the requirement**

TABLE 7.1: Requirements Format.

The format represents the following information:

- ID (example: FR_01). It unequivocally identifies the requirement so that it can later be traced and verified. In case the requirement is functional, the ID will begin with *FR* as in Functional Requirement. If it is a non-functional requirement, it will begin with *NFR*, as in non-functional requirement.
- Necessity: it specifies whether the requirement is important or not. If it is necessary, it means that the final application will not properly work without the completion of the requirement.
- Priority: we will use this field to identify if there are some requirements that must be fulfilled with a higher priority than others.
- Stability: in case the requirement is fixed and can not be changed, it will be considered as *stable*. Otherwise, it will be non-stable (if, for example, the requirement can or will be modified during the development of the application).
- Verifiability: it determines whether a requirement is able to be quantified, meaning that it can be proven that the requirement has been met.
- Status: the status of the requirement. In case there are some requirements that were proposed and later dismissed, it will be included in the *rejected* category.
- Type: whether the requirement is functional or not.

Now, we proceed to distinguish between functional and non-functional requirements, which we will see in Sections 7.1.1 and 7.1.2.

7.1.1 Functional Requirements

Functional requirements are those that describe the functionalities that the application must fulfill. Here is a list of all functional requirements, following the pre-established format seen in Table 7.1.

ID	FR_01
Necessity	Necessary
Priority	High
Stability	Stable
Verifiability	Verifiable
Status	Verified
Type	Functional
Name	Data Structure
Description	The application shall use a dequeue data structure to store heart rate information.

TABLE 7.2: Requirement FR_01.

ID	FR_02
Necessity	Necessary
Priority	High
Stability	Stable
Verifiability	Verifiable
Status	Verified
Type	Functional
Name	User Information
Description	The robot shall receive the age, gender and basal heart rate of the patient through standard input.

TABLE 7.3: Requirement FR_02.

ID	FR_03
Necessity	Necessary
Priority	High
Stability	Stable
Verifiability	Verifiable
Status	Verified
Type	Functional
Name	CA Communication
Description	The robot shall create CAs through the HRI manager module.

TABLE 7.4: Requirement FR_03.

ID	FR_04
Necessity	Necessary
Priority	High
Stability	Stable
Verifiability	Verifiable
Status	Verified
Type	Functional
Name	Telegram Communication
Description	The robot shall process requests to the Telegram API through the *telebot* module.

TABLE 7.5: Requirement FR_04.

ID	FR_05
Necessity	Necessary
Priority	High
Stability	Stable
Verifiability	Verifiable
Status	Verified
Type	Functional
Name	Constant Data
Description	The constant thresholds obtained in the statistical study shall be loaded from the file *constants.txt* in the specified path: <module_name>/ src/ medical_alert/ data/.

TABLE 7.6: Requirement FR_05.

7.1.2 Non-Functional Requirements

Non-functional requirements comprise the set of constraints of the system and quality standards from which the correct implementation of the application can be evaluated. Usability, scalability, performance and maintainability are factors to take into account when talking about non-functional requirements. We can see below the set of non-functional requirements for our application.

ID	NFR_01
Necessity	Necessary
Priority	High
Stability	Stable
Verifiability	Verifiable
Status	Verified
Type	Non-functional
Name	Operating System
Description	The application shall be compliant with the Ubuntu 16.04 LTS Operating System.

TABLE 7.7: Requirement NFR_01.

ID	NFR_02
Necessity	Necessary
Priority	High
Stability	Stable
Verifiability	Verifiable
Status	Verified
Type	Non-functional
Name	Bluetooth
Description	The platform where the application is executed shall count with a Bluetooth Low Energy (BLE) adapter.

TABLE 7.8: Requirement NFR_02.

ID	NFR_03
Necessity	Necessary
Priority	High
Stability	Stable
Verifiability	Verifiable
Status	Verified
Type	Non-functional
Name	Xiaomi Band Bluetooth
Description	The Xiaomi Mi Band 2 wearable device used for HRI shall be paired up with Mini Maggie prior to execution.

TABLE 7.9: Requirement NFR_03.

ID	NFR_04
Necessity	Necessary
Priority	High
Stability	Stable
Verifiability	Verifiable
Status	Verified
Type	Non-functional
Name	Xiaomi Band Battery
Description	The Xiaomi Mi Band 2 wearable device shall have, at least, 10% available battery.

TABLE 7.10: Requirement NFR_04.

ID	NFR_05
Necessity	Necessary
Priority	High
Stability	Stable
Verifiability	Verifiable
Status	Verified
Type	Non-functional
Name	Microphone Connection
Description	The platform where the application is executed shall have a microphone connected.

TABLE 7.11: Requirement NFR_05.

ID	NFR_06
Necessity	Necessary
Priority	High
Stability	Stable
Verifiability	Verifiable
Status	Verified
Type	Non-functional
Name	Internet Connection
Description	The platform where the application is executed shall have Internet connection available at all times.

TABLE 7.12: Requirement NFR_06.

ID	NFR_07
Necessity	Necessary
Priority	High
Stability	Stable
Verifiability	Verifiable
Status	Verified
Type	Non-functional
Name	Rate of Measurements
Description	The application shall make, at least, 10 requests requests per minute to the Xiaomi band.

TABLE 7.13: Requirement NFR_07.

ID	NFR_08
Necessity	Necessary
Priority	High
Stability	Stable
Verifiability	Verifiable
Status	Verified
Type	Non-functional
Name	Python Programming Language
Description	The application shall be developed in the Python programming language.

TABLE 7.14: Requirement NFR_08.

ID	NFR_09
Necessity	Necessary
Priority	High
Stability	Stable
Verifiability	Verifiable
Status	Verified
Type	Non-functional
Name	Python Version Support
Description	The application shall be compliant with the following Python versions: 2.6, 2.7 and 3.5.

TABLE 7.15: Requirement NFR_09.

7.2 Requirements Analysis

After all the functional and non-functional requirements have been presented, we make an analysis of all requirements, showing how each one of them, if their status is set as *verified*, have been verified. In our case, all of them have this status, so we have been able to verify that all of the requirements have been met. For this, we can observe two traceability matrix, one for functional requirements (Table 7.16) and another one for non-functional ones (Table 7.17). The set of use cases can be found in Sections 6.3.1, 6.3.2 and 6.3.3.

Requirement	1	2	3
FR_01	OK	OK	OK
FR_02	OK	OK	OK
FR_03	OK	OK	OK
FR_04	OK	OK	-
FR_05	OK	OK	OK

TABLE 7.16: Traceability Matrix for all functional requirements.

Requirement	1	2	3
NFR_01	OK	OK	OK
NFR_02	OK	OK	OK
NFR_03	OK	OK	OK
NFR_04	OK	OK	OK
NFR_05	OK	OK	OK
NFR_06	OK	OK	OK
NFR_07	OK	OK	OK
NFR_08	OK	OK	OK
NFR_09	OK	OK	OK

TABLE 7.17: Traceability Matrix for all non-functional requirements.

As we can observe, all requirements have been met by at least one of the three use cases; and in most of them, all three use cases.

Chapter 8

Planning

In this chapter, we will explain the task management followed throughout the development of this bachelor's work. As everything was decomposed initially into tasks, we can make an analysis of whether all tasks were finished at the end, and whether tasks took more time than initially expected. We will present two different GANTT diagrams for this: one for representing the initial set of tasks and the time that they had allotted, and another GANTT diagram for the time that it really took to complete all tasks.

Now, we will explain the set of tasks that we could decompose at the beginning of the project, with a set of subtasks in case we need them.

- **Task 0**: checking functional and non-functional requirements from scratch, to see what hardware we could need for the data collection process and the final functioning of the application. Also, analyzing ROS and Linux to check the functioning of modules in Mini Maggie. This involved learning about each component of Mini Maggie (ASR, TTS, HRI Manager and several other modules that we would need to communicate with).
- **Task 1**: investigation and analysis of Xiaomi Mi Band 2 devices. This task comprises everything from analyzing how much the battery lasted, as well as how we could extract information from the Xiaomi device (referring to the cryptography involved, as explained in Section 5.2.2.
  - **Task 1.1**: analyzing and testing previously created code from Leo Soares ([32]).
  - **Task 1.2**: performing a hexadecimal analysis and manually checking that the handshake is complete.
  - **Task 1.3**: requesting heart rate measurements and checking that the ROS node correctly got the information.
- **Task 2**: communicating ROS nodes (publisher and subscriber only) with Xiaomi band information (heart rate measurements). Also, creating additional functions to handle this data correctly.
- **Task 3**: creating an Object-Oriented Programming (OOP) paradigm to store this, and several other information that we would need in the final version of our robotic application.
- **Task 4**: planning a way to detect different kinds of activities based on the heart rate data we were collecting from the Xiaomi band.

  - **Task 4.1**: reading previous papers from heart rate variability depending on different activities.
  - **Task 4.2**: planning to make a quantitative, statistical study from scratch. Also, identifying the set of components we will need for the data collection process.

- **Task 5**: developing and testing an Android application for the data collection process.
  - **Task 5.1**: revisiting the Android activity lifecycle and the possible architectures with which to implement the application.
  - **Task 5.2**: design of all necessary activities, including auxiliary functions such as database structure and interface to communicate with activities.
  - **Task 5.3**: testing of the application after development, including a pre-testing phase with users from the Departamento de Sistemas y Automática, at UC3M.
  - **Task 5.4**: determining the publishing platform for the application. This section involved leveraging which would be the distribution method for test users. Firstly, the application was made available for Google Play, finally it was chosen to be delivered via Google Drive individually.
  - **Task 5.5**: application obfuscation, so that the contents of the application would be protected from reverse engineering and depackaging.

- **Task 6**: performing the data collection process.
  - **Task 6.1**: gathering a set of 36 initial test users.
  - **Task 6.2**: distributing the application, installing it in the test user's mobile devices, check for platform-dependent errors (such as Samsung devices), and attempting to fix API level errors in test user's mobile devices to install the application.
  - **Task 6.3**: supporting test users during the 3-day period of their study. From meeting with them in case batteries were depleted, to explaining them, in case they were confused, about which activity would correspond to which category (to indicate in the Android application).
  - **Task 6.4**: recovering hardware and resetting the mobile devices from test users to the original state they were three days ago.
  - **Task 6.4**: collecting consent form, user satisfaction forms and usability forms from users.

- **Task 7**: analyzing all data.
  - **Task 7.1**: discarding all users whose data was corrupt or unusable for some reasons (see Section 3.6).
  - **Task 7.2**: analyzing the final set of users that we had, by categorizing them by gender and age. For that, we made use of the consent form from users to recover their data. With this data, make an initial analysis of our initial set of data (see Section 4.1.1).
  - **Task 7.3**: cross-referencing all data from users by separating all their data in different files programatically (with Python).

  - **Task 7.4**: cleaning data and preparing it for Machine Learning (ML) analysis.
  - **Task 7.5**: analyzing static data for age and gender and extract the static thresholds that would be later used for detecting dangerous heart rates.

- **Task 8**: developing ML techniques to analyze data, with the already prepared and cleaned data, and extract conclusions from the results.
- **Task 9**: iterate several times to try and improve the ML behavior. Try with several different classifying algorithms, apart from Random Forest.
- **Task 10**: with all this data, create the robotic application (with static thresholds) and test the functionality.
  - **Task 10.1**: include static thresholds and the Random Forest classifier.
  - **Task 10.2**: predict new data from the Xiaomi band in real time through ML techniques.
  - **Task 10.3**: include own-developed modules (such as Telegram, Email) and test their functionality.
  - **Task 10.4**: test the proper functioning of the robotic application.
  - **Task 10.5**: create three different use cases for the application, to verify that the application works in all possible cases.
- **Task 11**: drafting of the report.
- **Task 12**: review of the report, performing necessary corrections through several iterations for each chapter.

Now that we have quantified all tasks that were involved, we proceed to show the first GANTT diagram, which corresponds to the predicted schedule for all tasks throughout this bachelor's work. We can observe this GANTT diagram in Figure 8.1.

Finally, we can make a comparison from the original in Figure 8.1 and the real one, which can be found in Figure 8.2. In conclusion, the development of all requirements and tasks was much longer than we predicted, and this bachelor's work took more than recommended by guidelines.

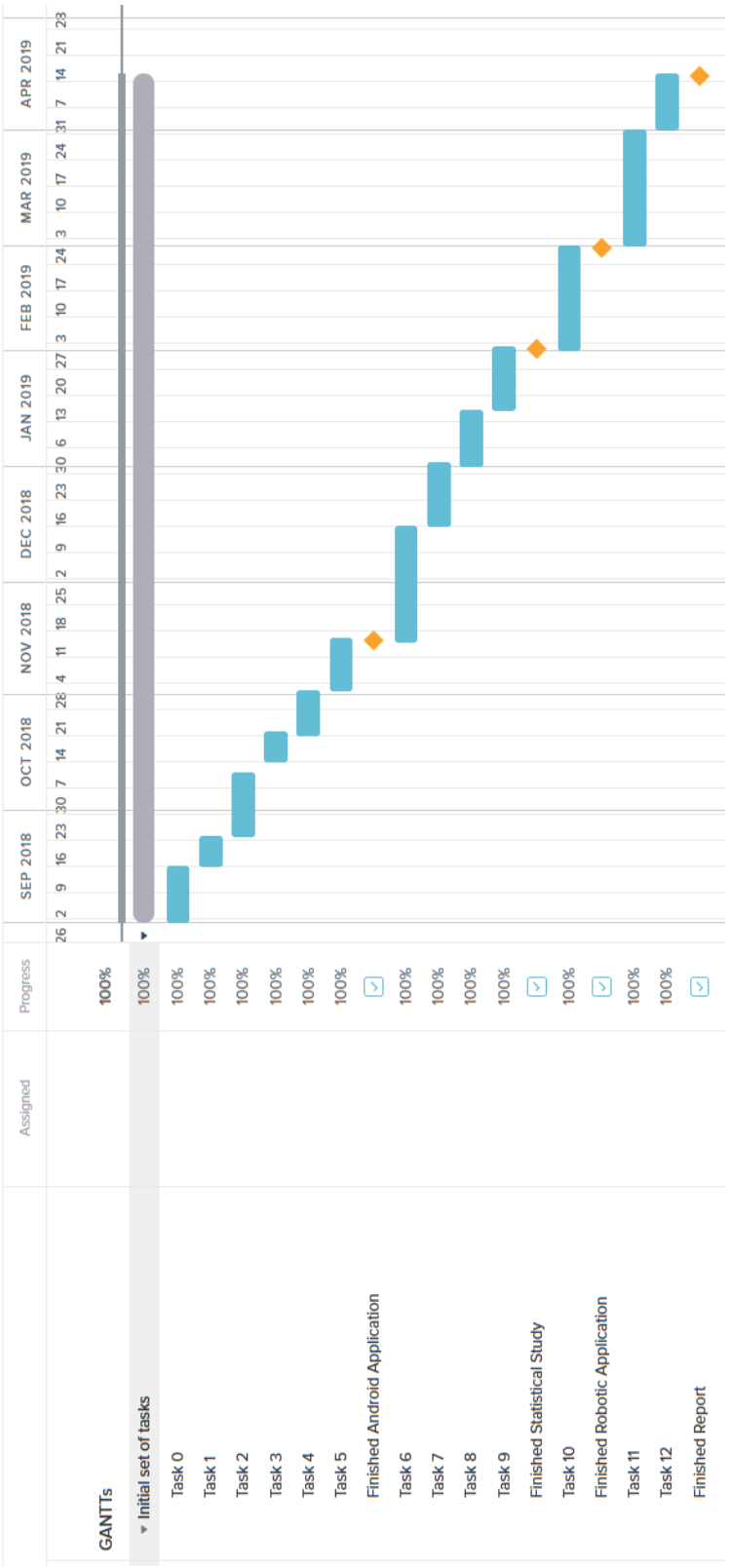


FIGURE 8.1: GANTT diagram for the initial set of tasks and the time that they had allotted.

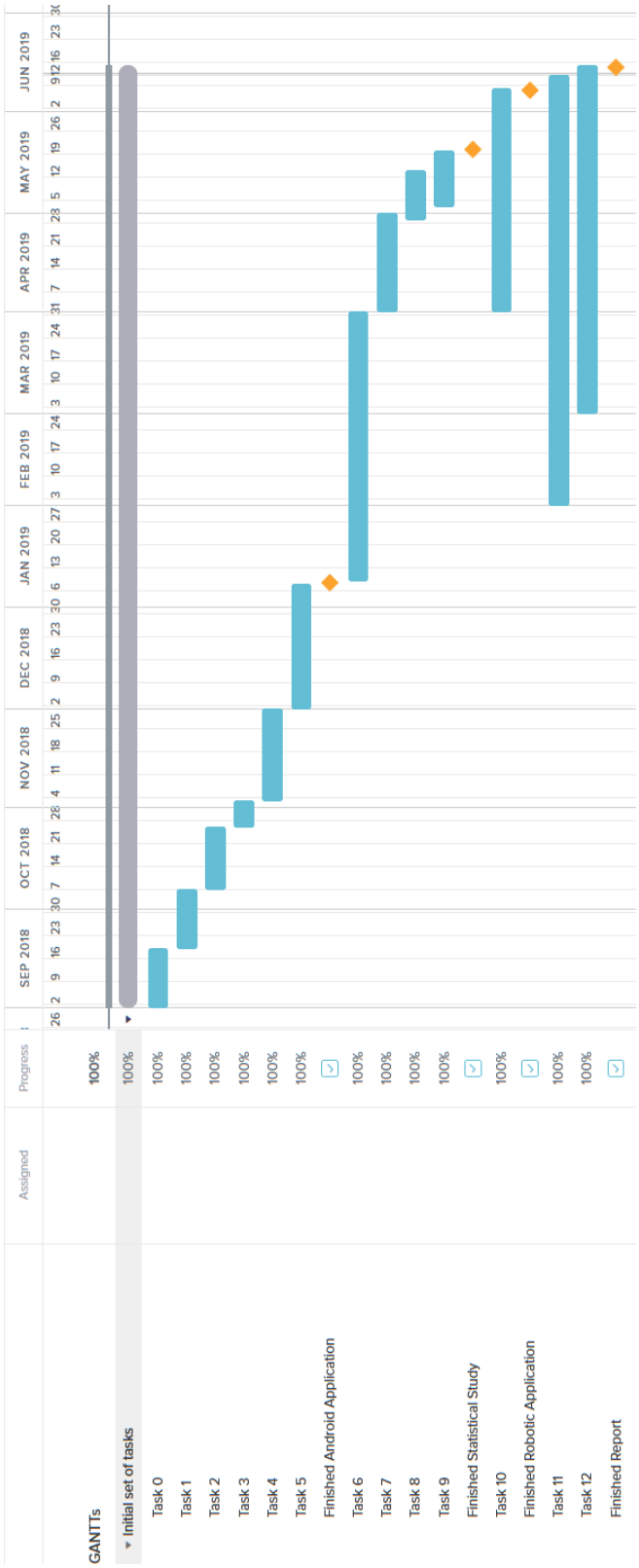


FIGURE 8.2: GANTT diagram for the real set of tasks and the time that they really took.

As observable in Figures 8.1 and 8.2, Task 6 took much longer than expected. This was caused because we predicted that the time it would take to perform the data collection process turned out to be much more than expected. The process of collecting data ended up being very time consuming because we needed to meet with the test users at the beginning and ending of their 3-day period. Due to scheduling issues, from our or their side, the data collection processes ended up being approximately 2 to 3 times longer than initially calculated.

Additionally, Task 11 and 12 took longer too, because in reality, the drafting and reviewing of chapters in the report were made iteratively and was started much earlier, doing these tasks simultaneously during the whole data collection process.

Chapter 9

Socio-Economic Environment

In this chapter, we will observe the set of socio-economic factors that were involved in this bachelor's work. First of all, we will make a prediction of the cost of the development of the project, in Section 9.1. Finally, we will predict the socio-economic impact that the successful production of this bachelor's work would have, in Section 9.2.

9.1 Budgeting

Taking into account the information from [34], we make a preidction about the standard cost of the project, deriving from the total number of hours required to complete this bachelor's work. Taking into account that each week, in Spain, is comprised by 48 hours a week, and there are four weeks in a fiscal month, the total number of billable hours in a month is 192. We can observe the total cost in Table 9.1.

Number of Hours	Salary per Month ([34])	Nr. Billable Months	Total Cost
720	1215.90 EUR	3.75	4559.63 EUR

TABLE 9.1: Project Total cost, based on the number of hours used for the development of this bachelor's work. The number of months is calculated as the number of hours devoted to this bachelor's work divided by 192, which is the number of billable hours per month.

This cost does not include the software and hardware costs. In our case, as the development of all modules of this project has been made with open-source and free tools, the software cost is 0 EUR. However, we need to make a better analysis of the hardware cost.

9.1.1 Hardware Costs

In regards to the hardware cost, we need to consider several factors. First, we must consider the hardware purchased to collect data and test our application. In our case, we bought 3 Xiaomi Mi Band 2 devices, each with a standard price of at the time of purchase. The standard cost for each one of the Xiaomi devices, at the time of purchase, was 19.90 EUR.

Additionally, we need to consider the use of the robot Mini Maggie, used for the development and testing of the software product explained in Chapter 5. The total cost of Mini Maggie is 2600 EUR. For utilizing the robot, we need to calculate the *amortization factor* for the use of Mini Maggie (not its purchase).

Taking into account that the standard cost is 2600 EUR, the useful life of Mini Maggie is 84 months, and we have used Mini Maggie for a total of 9 months for the development of this bachelor's work, the amortization factor cost for Mini Maggie is observed in Table 9.2.

Months of Use	Useful Life	Mini Magie Cost	Amortization Factor
9	78	2600 EUR	278.57 EUR

TABLE 9.2: Calculation of the amortization factor cost for Mini Maggie. The amortization factor is the total cost multiplied by the percentage of useful life employed in the development of this bachelor's work (9/84).

Therefore, the total hardware cost is the sum of both the amortization factor cost plus the cost of the three Xiaomi Mi Band 2 devices. We can observe the total hardware cost in Table 9.3.

Item	Units	Cost per Unit	Total Cost
Xiaomi Mi Band 2	3	19.90EUR	59.70EUR
Mini Maggie Amortization Cost	1	258.57EUR	258.57EUR
Total	1	1	318.29EUR

TABLE 9.3: Calculation of the amortization factor cost for Mini Maggie. The amortization factor is the total cost multiplied by the percentage of useful life employed in the development of this bachelor's work (9/84).

Finally, summing the hardware costs from Table 9.3 and the standard costs from Table 9.1, we obtain a total project cost of **4877.92 EUR**.

9.2 Socio-Economic Impact

The impact of the development of the robotic software module would be primarily social. As the development of this module has no economic impact, as the robot will have the same base price regardless of the number of software modules included inside, the economic impact is null.

However, the development of this module attempts to improve the social perception of robotics by offering a better Human-Robot Interaction (HRI) for Mini Maggie. Also, several of the used tecniques can be used in the future to monitor vital information of users to allow a better control of this information by the robot. The development of a future robot-based alert system is very possible.

Chapter 10

Conclusions

In this chapter, we make a summary about the work that has been done in this bachelor's work, what has been achieved and what has not. Additionally, we will explain the future work related to all that we have done, in Section 10.2.

10.1 What has been done

From the initial set of objectives in Section 1.3, we see the following objectives of this bachelor's work.

1. Implement a social robot able to transparently communicate with a user that is in distress and needs immediate help.
2. Develop a static algorithm to detect subtle differences in a patient's heart rate, to determine if a patient is in danger or needs assistance.
3. Through the use of Machine Learning algorithms, generate a classification model that can conclude, in real time, the predicted activity a user is performing at any given time, based on the user's past and current heart rate measurements.
4. Being able to help others, in a cheap and effective way, with a tool that can be worn by any patient to log heart rate information.

From this list, we can say the following:

1. The social robot was able to transparently communicate with any user through the development of our own module, explained in Section 5.2, and the use of auxiliary third-party modules as explained in Section 5.3.
2. Through the use of a statistical, well-organized study of heart rate information from 24 final users, we were able to determine a set of thresholds that determined the average for each one of the daily activities in a user (physical exercise, others, walking, sleeping and resting). We were able also to categorize this data into three different age groups (0 to 30 years old, 30 to 60, and more than 60) and two different genders (male and female) to improve the preciseness of our thresholds. We can see the results from this statistical study in Sections 4.1.4 and 4.1.3.
3. Developing a machine learning model through a Random Forest classifier, we were able to predict, with a precision of 51%, the current activity of a user based on his/her history of heart rates. However, the precision of the machine learning model is not enough, and we believe that with a better statistical study,

we will be able to improve this statistic in the future, until a much higher prediction rate of about 90%. In Section 6.3.3 we can see, in practice, how the implementation of this machine learning model can improve HRI with Mini Maggie.

4. Helping others with our own developed module was also achieved. Through a seamless interaction with the robot through HRI, we have been able to improve the interaction of Mini Maggie with all users, introducing additional communication patterns that can improve the interaction between any user and Mini Maggie. In Sections 6.3.1, 6.3.2 and 6.3.3, we can see in practice all the quality of interaction between Mini Maggie and a test user.

10.2 Future Work

The set of guidelines for future work with regards to this bachelor's work are:

- Improve the machine learning model, by improving the quality of the data collection process in the statistical study. This is the purpose of a future *scientific paper* linked to this bachelor's work. It will preliminarily consist on requesting heart rate information from the Xiaomi Mi Band 2 wearable device much more frequently than the GadgetBridge application would allow us to request; thus improving the amount of measurements per minute and subsequently improving the overall quality of data for each activity category.
- Increase the number of test users to perform a more accurate statistical study and data collection processes, with at least 30 users for each age and group (e.g. 30 males and 30 females under 30 years of age).
- Include additional information from users so that better comparisons can be made between them. Also, include test users with congenital heart diseases and study the variability between their heart rate information and the information from healthy users.
- Increase the number of Xiaomi band devices to a much higher amount, e.g. 10-20 devices so that the data collection process can be made much more efficiently, which was one of the drawbacks from this bachelor's work.
- Consider the possibility to make the data collection process with other devices, such as Xiaomi Mi Band 3 devices; as they have a better reliability and lower standard deviation in regards to heart rate measurements (Section 6.1).

has a purpose other than being the medical system itself (as the Xiaomi Mi Band 2 hardware is used, the device can be used as a watch, as well as including other features that Xiaomi Mi Band 2's specifications allow, such as a measurement for the number of steps and calories used throughout the day).

Bibliography

[1] Joel E Cohen. "Human population: the next half century". In: *science* 302.5648 (2003), pp. 1172–1175.

[2] Jeanette Takamura and Bob Williams. "Informal caregiving: Compassion in action". In: *Department of Health and Human Services: Washington, DC, USA* (1997).

[3] World Health Organization. *Falls*. http://www.who.int/mediacentre/factsheets/fs344/en/. Accessed on June 2019. 2016.

[4] Klaas A Hartholt, Ed F Van Beeck, and Tischa JM Van Der Cammen. "Mortality from falls in Dutch adults 80 years and older, 2000-2016". In: *Jama* 319.13 (2018), pp. 1380–1382.

[5] Stanley L Cruitt, Sidney Chan, and Jaroslav V Tichy. *Portable programmable medical alert device*. US Patent 6,934,220. 2005.

[6] Brian Carrier and Kathleen A Maier. *Emergency alert and security apparatus and method*. US Patent 5,195,126. 1993.

[7] Michael Sasha John and David R Fischell. *Medical alarm and communication system and methods*. US Patent 8,002,701. 2011.

[8] Liang Liu et al. "Automatic fall detection based on Doppler radar motion signature". In: *2011 5th International Conference on Pervasive Computing Technologies for Healthcare (PervasiveHealth) and Workshops*. IEEE. 2011, pp. 222–225.

[9] C Barr Taylor et al. "Ambulatory heart rate changes in patients with panic attacks." In: *The American journal of psychiatry* (1986).

[10] Jack M Gorman and Richard P Sloan. "Heart rate variability in depressive and anxiety disorders". In: *American heart journal* 140.4 (2000), S77–S83.

[11] Opensource.org. *The 3-Clause BSD License*. https://opensource.org/licenses/BSD-3-Clause. Accessed on June 2019.

[12] IMSERSO and FEMP. "Imserso. Instituto de Mayores y Servicios Sociales. Normas Generales del Servicio de Teleasistencia Domiciliaria. Programa de Teleasistencia IMSERSO-FEMP. Ministerio de Sanidad y Política Social". In: (Oct. 1999).

[13] Li Zhengzhou et al. "Human body fall detection alarm device based on multiple sensors". In: (2012). CN Patent=102,800,170A.

[14] Diane K Barker. *Medical information appliance*. US Patent 6,560,165. 2003.

[15] David Zarchan. *Medical reminder system and messaging watch*. US Patent 6,075,755. 2000.

[16] Hoi Tung et al. "A mobility enabled inpatient monitoring system using a ZigBee medical sensor network". In: *Sensors* 14.2 (2014), pp. 2397–2416.

[17] Miquel Domènech. "Teleasistencia publica en España: consideraciones de sus efectos en los servicios sociales y sanitarios". In: ().

[18] Kuan-Yu Chen, Hsing-Wang Chang, and Feng-Hsiung Wu. *Waterproof structure and portable electrical apparatus using the same*. US Patent 8,089,757. 2012.

[19] Dan Roman. "Introduction to IEC 60335—Household and similar electrical appliances—Safety". In: *2015 IEEE Symposium on Product Compliance Engineering (ISPCE)*. IEEE. 2015.

[20] International Electrotechnical Commission et al. "IEC 60529: degrees of protection provided by enclosures (IP code)". In: *See Appendix* (1989).

[21] Jin-Shyan Lee, Yu-Wei Su, Chung-Chou Shen, et al. "A comparative study of wireless protocols: Bluetooth, UWB, ZigBee, and Wi-Fi". In: *Industrial electronics society* 5 (2007), pp. 46–51.

[22] Chatschik Bisdikian et al. "An overview of the Bluetooth wireless technology". In: *IEEE Commun Mag* 39.12 (2001), pp. 86–94.

[23] Joshua Wright. *Security Laboratory: Wireless Security. Dispelling Common Bluetooth Misconceptions*. https://www.sans.edu/cyber-research/security-laboratory/article/bluetooth. Accessed on June 2019.

[24] Joseph Shinar. *Organic light-emitting devices: a survey*. Springer Science & Business Media, 2013.

[25] Andreas Shimokawa. *Mi Band 2 Firmware Update*. https://github.com/Freeyourgadget/Gadgetbridge/wiki/Mi-Band-2-Firmware-Update. Accessed on June 2019.

[26] Bjarne Stroustrup. *The C++ programming language*. Pearson Education India, 2000.

[27] Morgan Quigley et al. "ROS: an open-source Robot Operating System". In: *ICRA workshop on open source software*. Vol. 3. 3.2. Kobe, Japan. 2009, p. 5.

[28] *Understand the Activity Lifecycle | Android Developers*. https://developer.android.com/guide/components/activities/activity-lifecycle. Accessed on June 2019.

[29] Hirofumi Tanaka, Kevin D Monahan, and Douglas R Seals. "Age-predicted maximal heart rate revisited". In: *Journal of the american college of cardiology* 37.1 (2001), pp. 153–156.

[30] A Mesquita et al. "The maximum heart rate in the exercise test: the 220-age formula or Sheffield's table?" In: *Revista portuguesa de cardiologia: orgao oficial da Sociedade Portuguesa de Cardiologia= Portuguese journal of cardiology: an official journal of the Portuguese Society of Cardiology* 15.2 (1996), pp. 139–44.

[31] Michael M Dehn and Robert A Bruce. "Longitudinal variations in maximal oxygen intake with age and activity." In: *Journal of Applied Physiology* 33.6 (1972), pp. 805–807.

[32] Leo Soares. *Mi Band 2, Part 1: Authentication*. https://leojrfs.github.io/writing/miband2-part1-auth/. Accessed on June 2019.

[33] Junqing Xie et al. "Evaluating the validity of current mainstream wearable devices in fitness tracking under various physical activities: comparative study". In: *JMIR mHealth and uHealth* 6.4 (2018).

[34] *Bases y tipos de cotización 2019*. http://www.seg-social.es/wps/portal/wss/internet/Trabajadores/CotizacionRecaudacionTrabajadores/36537?changeLanguage=es. Accessed on June 2019.

Appendix A

Study Forms

A.1 English Consent Form

Statistical Study – Consent Form

Ignacio Guillermo Martínez Rincón
+34689275855 – 23caj23@gmail.com

Title of Study: Smartwatch-Based User Monitoring for Human-Robot Interaction

Investigator: Ignacio Guillermo Martínez Rincón, Department of Automation Systems at Universidad Carlos III de Madrid

Description of the Study

- If you agree to take part in this study, you will be asked to:
  - Download an Android application used for logging activities throughout the day. You will need to access the application and note which activity you're going to do, and for how long. You must do this as frequently as possible to help me analyze data afterwards.
  - Always wear a Mi Band 2 device for the whole duration of the study. This will also mean during the shower, and while in bed (sleeping / taking naps).
  - Accurately note the time periods where a non-sedentary physical activity has occurred.
    - For example, if you went jogging for 1 hour on Tuesday, I'll need you to access the Android application and note 'Physical exercise', with a 1-hour duration. This will greatly help me analyze the huge amounts of data I'll receive from your wearable device.
    - The more descriptive and thorough you are, the more you will help me.
  - Inform me of any problems with the wearable device as soon as possible – e.g. battery failure, non-responsive device, problems with the Android application…
  - At the end of the study, I will need to meet with you to recover the wearable device, as well as to download the database file where the Android application stored all the data you provided.
- The general duration of your involvement in the study will be 3 days.

Confidentiality

- Your name will not be used in the study, you will be identified by an ID and will be identified as such when your data is analyzed and shown.
- Therefore, this study is anonymous. After the data mining period, analysis and possible display of some of the confidential data, all the databases extracted from the Mi Band 2 wearable devices will be deleted.

FIGURE A.1: Consent Form (English Version), Page 1

Signature

If you sign this consent, you hereby agree to all the above-mentioned terms, promising that you will accurately provide me with truthful information, and that you agree to the display and use of your confidential data.

Finally, I will ask you to complete a small survey to get a better profiling about the data that I'm going to process:

----- **Questionnaire** -----

I am employed _____ (Yes/No)

I exercise _____ times a week

I suffer from a congenital heart disease _____ (Yes/No)

Relative(s) have suffered from heart attacks in the past _____ (Yes/No)

I smoke regularly _____ (Yes/No)

I am _____ years old

I was born a _____ (Male / Female)

Participant's Signature: _____

Researcher's Signature: _____

Date:

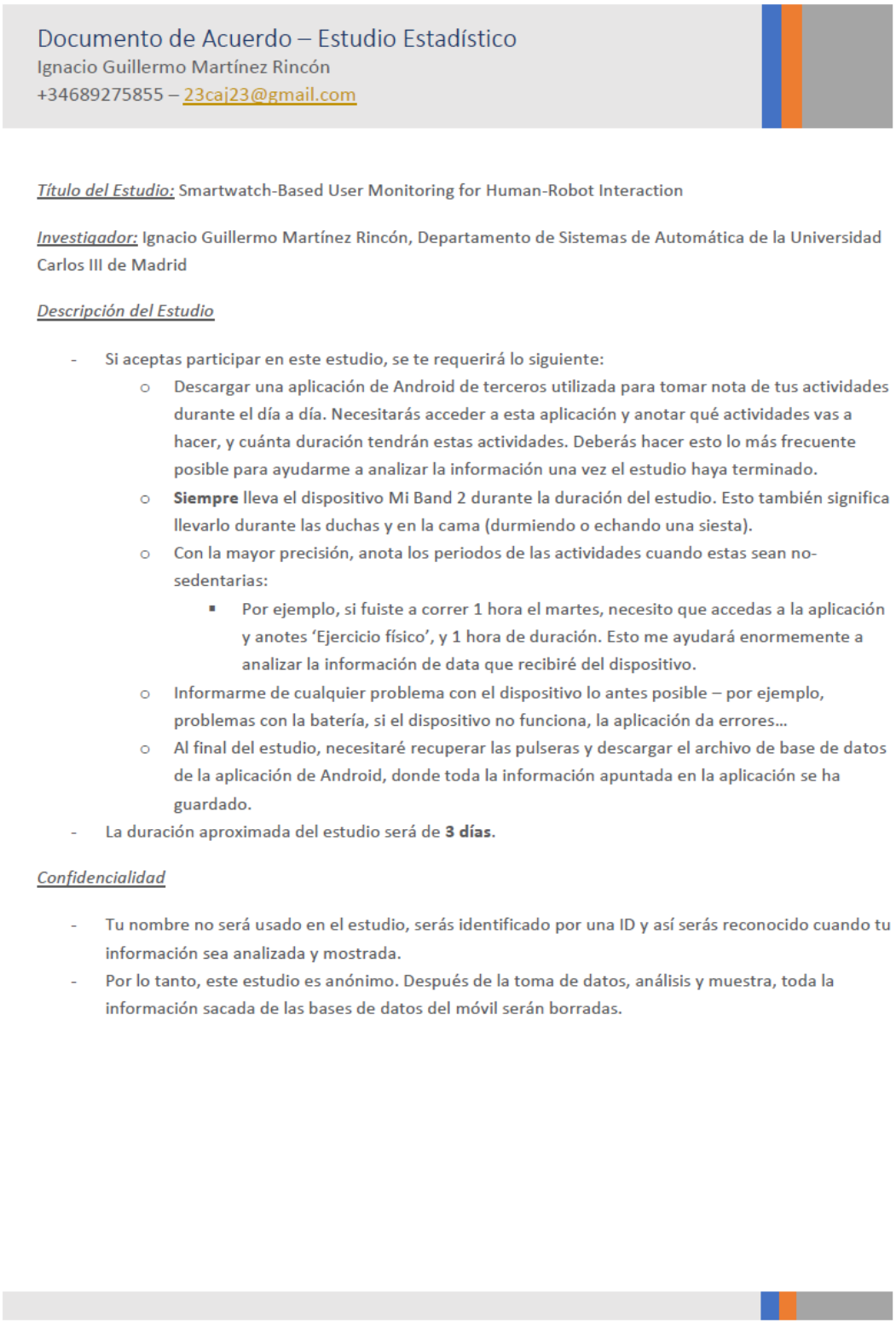
ID:

Device HWID:

Period of study:

FIGURE A.2: Consent Form (English Version), Page 2

A.2 Spanish Consent Form



Documento de Acuerdo – Estudio Estadístico

Ignacio Guillermo Martínez Rincón
+34689275855 – 23caj23@gmail.com

Título del Estudio: Smartwatch-Based User Monitoring for Human-Robot Interaction

Investigador: Ignacio Guillermo Martínez Rincón, Departamento de Sistemas de Automática de la Universidad Carlos III de Madrid

Descripción del Estudio

- Si aceptas participar en este estudio, se te requerirá lo siguiente:
  - Descargar una aplicación de Android de terceros utilizada para tomar nota de tus actividades durante el día a día. Necesitarás acceder a esta aplicación y anotar qué actividades vas a hacer, y cuánta duración tendrán estas actividades. Deberás hacer esto lo más frecuente posible para ayudarme a analizar la información una vez el estudio haya terminado.
  - **Siempre** lleva el dispositivo Mi Band 2 durante la duración del estudio. Esto también significa llevarlo durante las duchas y en la cama (durmiendo o echando una siesta).
  - Con la mayor precisión, anota los periodos de las actividades cuando estas sean no-sedentarias:
    - Por ejemplo, si fuiste a correr 1 hora el martes, necesito que accedas a la aplicación y anotes 'Ejercicio físico', y 1 hora de duración. Esto me ayudará enormemente a analizar la información de data que recibiré del dispositivo.
  - Informarme de cualquier problema con el dispositivo lo antes posible – por ejemplo, problemas con la batería, si el dispositivo no funciona, la aplicación da errores...
  - Al final del estudio, necesitaré recuperar las pulseras y descargar el archivo de base de datos de la aplicación de Android, donde toda la información apuntada en la aplicación se ha guardado.
- La duración aproximada del estudio será de **3 días**.

Confidencialidad

- Tu nombre no será usado en el estudio, serás identificado por una ID y así serás reconocido cuando tu información sea analizada y mostrada.
- Por lo tanto, este estudio es anónimo. Después de la toma de datos, análisis y muestra, toda la información sacada de las bases de datos del móvil serán borradas.

FIGURE A.3: Consent Form (Spanish Version), Page 1

Firma

Si firmas este acuerdo, estás de acuerdo con todos los términos previamente mencionados, prometiendo que la información que me des será lo más veraz posible, y que aceptas la muestra y el uso de la información confidencial previamente mencionada.

Finalmente, te pediré que, antes de comenzar con el estudio, respondas un cuestionario que me permitirá hacer un perfil sobre la información que procesaré:

---- **Cuestionario** ----

Tengo trabajo _____ (Yes/No)

Hago ejercicio _____ veces a la semana

Sufro de problemas de corazón heredados _____ (Sí/No)

Miembros de mi familia han sufrido paradas cardio-respiratorias en el pasado _____ (Sí/No)

Fumo con frecuencia _____ (Sí/No)

Tengo _____ años

Nací _____ (Hombre / Mujer)

Firma del Participante: _____

Firma del Investigador: _____

Fecha:

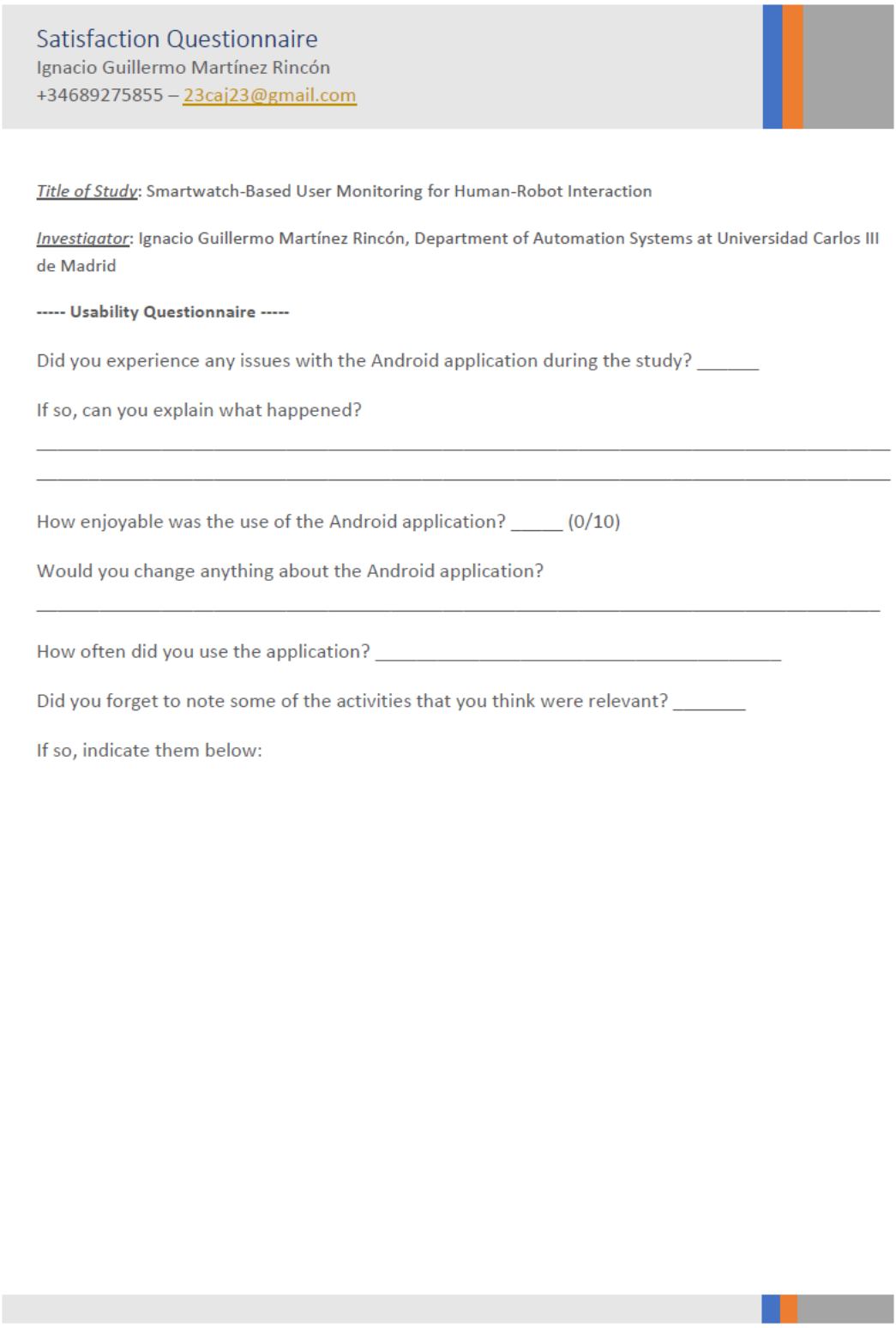
ID:

HWID del dispositivo:

Periodo de estudio:

FIGURE A.4: Consent Form (Spanish Version), Page 2

A.3 English Satisfaction Form



Satisfaction Questionnaire
Ignacio Guillermo Martínez Rincón
+34689275855 – 23caj23@gmail.com

Title of Study: Smartwatch-Based User Monitoring for Human-Robot Interaction

Investigator: Ignacio Guillermo Martínez Rincón, Department of Automation Systems at Universidad Carlos III de Madrid

----- Usability Questionnaire -----

Did you experience any issues with the Android application during the study? _____

If so, can you explain what happened?

_____

_____

How enjoyable was the use of the Android application? _____ (0/10)

Would you change anything about the Android application?

_____

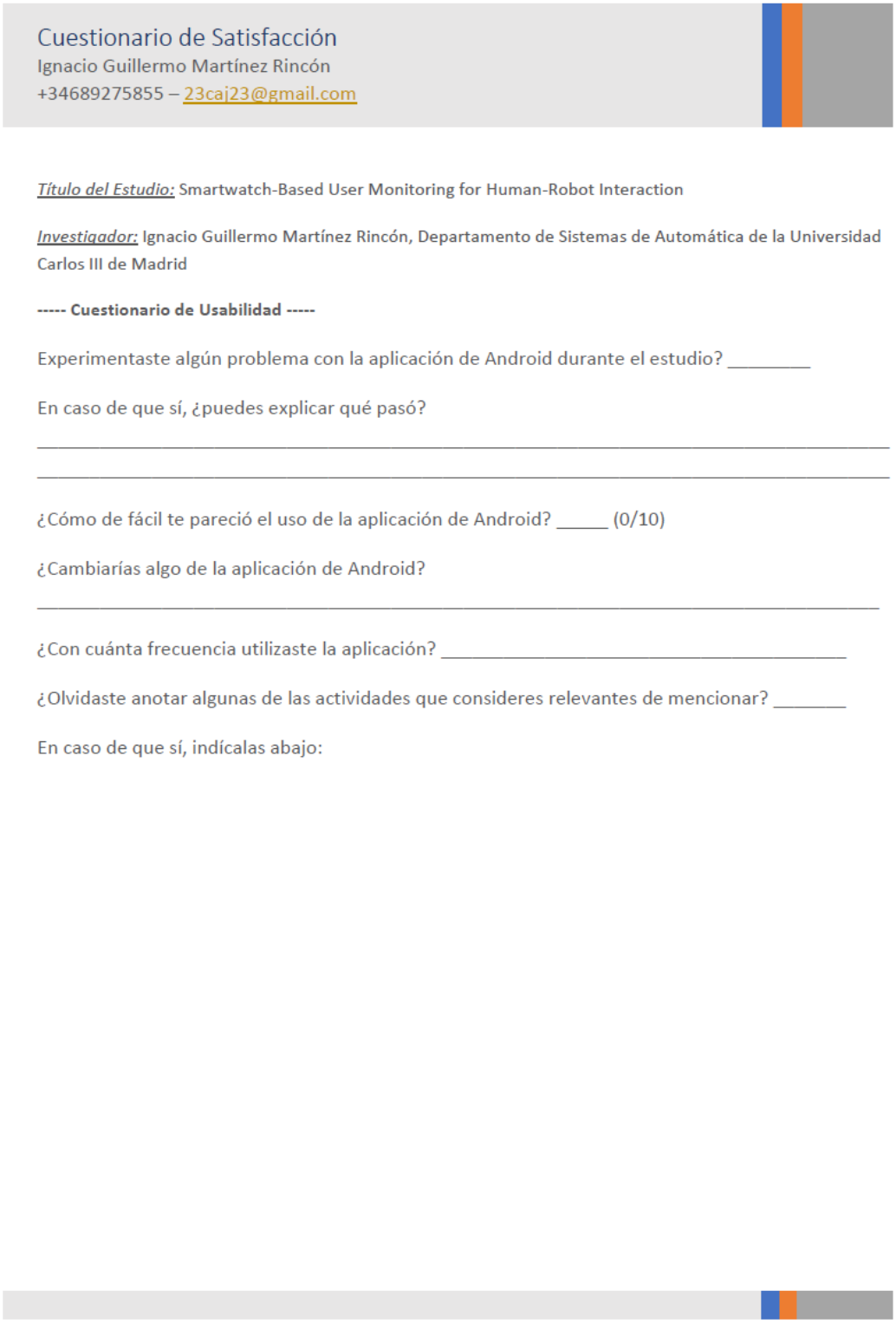
How often did you use the application? _____

Did you forget to note some of the activities that you think were relevant? _____

If so, indicate them below:

FIGURE A.5: Satisfaction Form (English Version)

A.4 Spanish Satisfaction Form



Cuestionario de Satisfacción

Ignacio Guillermo Martínez Rincón
+34689275855 – 23caj23@gmail.com

Título del Estudio: Smartwatch-Based User Monitoring for Human-Robot Interaction

Investigador: Ignacio Guillermo Martínez Rincón, Departamento de Sistemas de Automática de la Universidad Carlos III de Madrid

----- Cuestionario de Usabilidad -----

Experimentaste algún problema con la aplicación de Android durante el estudio? _____

En caso de que sí, ¿puedes explicar qué pasó?

_____

_____

¿Cómo de fácil te pareció el uso de la aplicación de Android? _____ (0/10)

¿Cambiarías algo de la aplicación de Android?

_____

¿Con cuánta frecuencia utilizaste la aplicación? _____

¿Olvidaste anotar algunas de las actividades que consideres relevantes de mencionar? _____

En caso de que sí, indícalas abajo:

FIGURE A.6: Satisfaction Form (Spanish Version)

Appendix B

Android Activity Lifecycle

Associated to all these states, there is a set of callbacks/functions that model each of the states:

- onCreate(). This method is executed once, when the system instantiates the activity. The activity is considered as *created*. This method is executed, therefore, only one for the entire lifecycle of the activity, regardless of how many times we enter the application. Every time the user accesses an activity after launching the application, the onCreate() function is called. This means that, as long as the application's process is alive and stored in main memory, onCreate() will be only called once.

- onStart(). This method causes an activity to acquire the *Started* state. This function is derived from the execution of the onCreate() function. This means that all the creation and initialization procedures made in the *Created* state of the activity has been completed. Once the activity has finished the execution of this callback, it does not stay resident in the *Started* state. It calls the subsequent onResume() function.

- onResume(). When this method is called, the activity acquires the state *Resumed*. This state is resident, unlike all other states previously mentioned. It means that, as long as the user executing the application stays on the activity, the onResume() callback will endlessly executing. However, when there is an event that would stop the *Resumed* state, for example, accessing a different activity on the application, the application leaves the *Resumed* state and the onPause() callback is called.

- onPause(). When this method is called, the activity acquires the state *Paused*. This callback is temporary and acts as a preemptive measure to inform the activity that the user is exiting its execution, and indicates that the activity is no longer on foreground. When the activity moves to the *Paused* state, system resources start being freed, such as sensors (GPS, accelerometers, camera, or any other I/O that was used in the activity), to prevent battery draining. The execution of this callback is very brief and may or may not be sufficient to store important information persistently. Therefore, most of these saving operations are and shall be performed, according to Android developer guidelines, in the onStop() callback. Finally, the user may return to the *Resumed* state in case that the user returns to the activity fast enough.

- onStop(). In this method, the activity is completely stopped and therefore acquires the *Stopped* state, and it continues, as mentioned, to perform store operations needed after the execution of the activity. Such activities can be, for

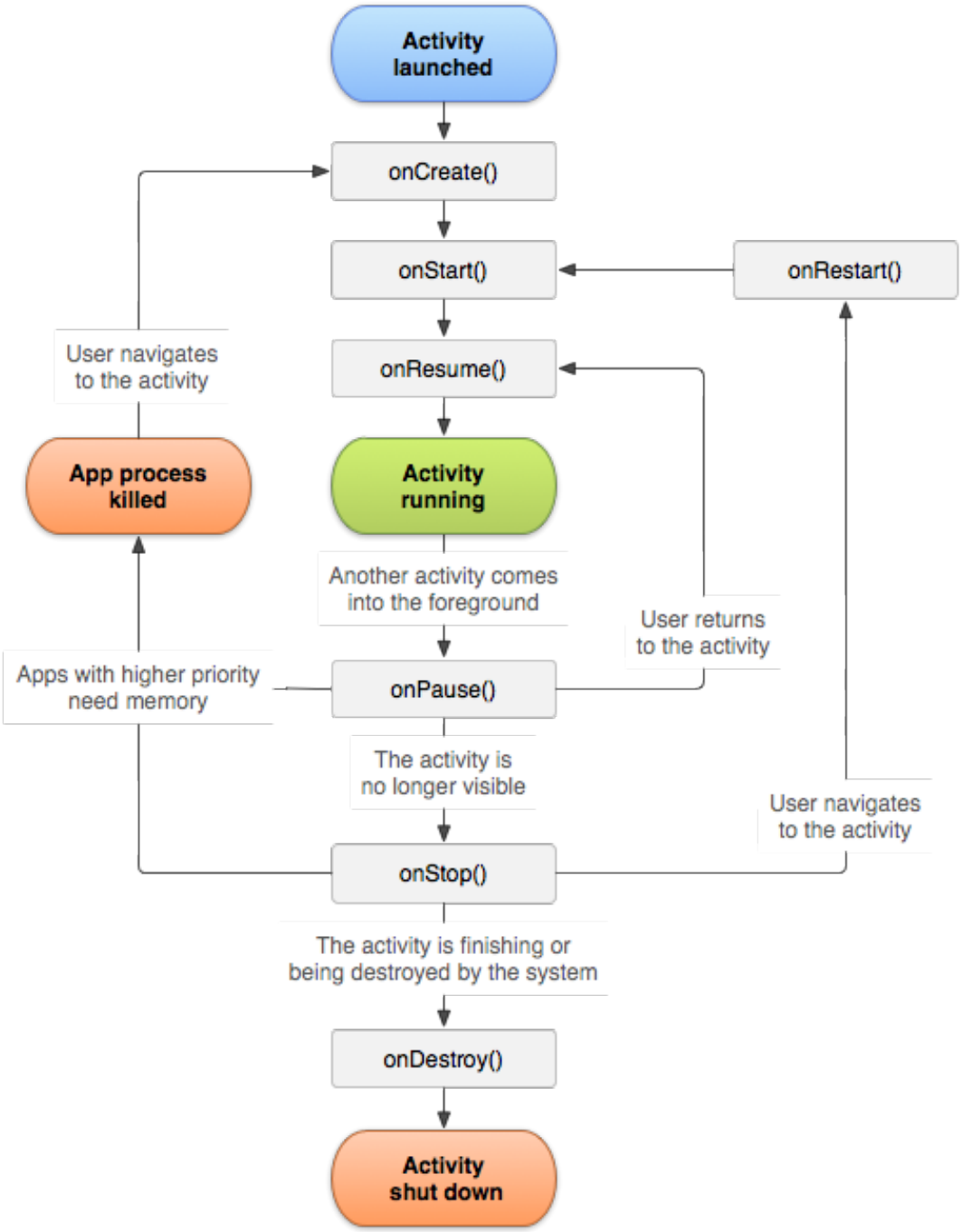


FIGURE B.1: Android Activity Lifecycle ([28]).

example, storing logging information in a database about the user's activities. In case of an activity being in the *Stopped* state, two things may occur:

1. The application is destroyed by the system due to inactivity, or a change in the mobile device's orientation or any other mobile configuration modification that affects the activity's properties. In this case, the onDestroy() callback is invoked.
2. The user resumes the use of the activity by re-accessing it. In this case, the onStart() callback is invoked.

- onDestroy(). In this case, the activity acquires the *Destroyed* state and is shutdown until the application re-accesses the activity. This may happen because the finish() system call is called upon the activity, or because the user has completely dismissed the activity. Also, if the layout of the activity is affected by an event, such as an orientation change in the device (for example), the activity is destroyed until re-launched with its corresponding new layout.

In the case of the Android application, all activities implicitly use all methods explained in Appendix B. However, some methods are overriden to extend the functionality of activities into something more concrete. In this case, the application modifies the behavior of the onCreate() function in all activities, leaving all other functions with the standard Android implementation. Therefore, the explanation of all unused functions and the activity lifecycle of the Android application, which is deemed necessary to understand how activities work internally, were left in Annex B for that purpose.
Referring to the onCreate() function, the implementation of all activities are made in this method. The Android application mostly consists on navigation buttons for the user. It was designed and simple and accessible as possible, to make it very usable. Also, the database operations (used to store the selections made by the user) are not made when the activity is stopped, as it normally happens. Database operations happen when the user press a *confirm* button to commit the user's selection in the GUI. If the user does not press this button, nothing is stored in the database. Therefore, the activity must be in foreground and the user must be actively interacting with this activity through the GUI for database actions to occur.

Finally, note that, activities do not share information between each other. However, we chose the design of the application to consider all activities as modular and independent between each other, apart from the buttons that interconnect them and allow navigation throughout the application.

The set of activities in the Android application are:

- MainActivity.java
- Homepage.java
- Duration.java
- DatabaseHelper.java
- Data.java

Appendix C

Sequence Diagrams

In this Annex, we can observe the set lower-level, function-specific sequence diagrams for the Android application as explained in Section 3.3.3

C.1 Homepage

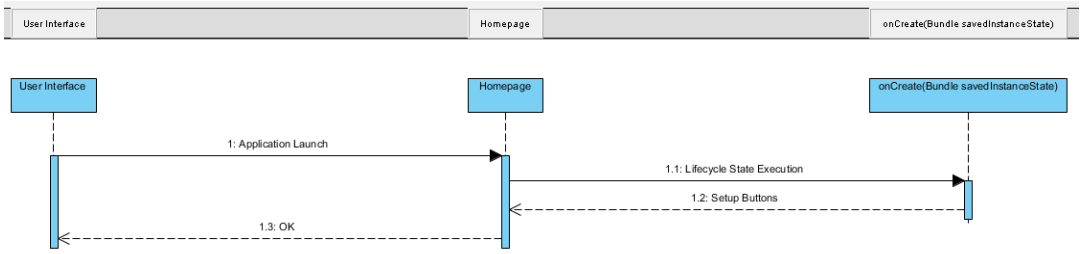


FIGURE C.1: Sequence Diagram of Homepage class, method onCreate().

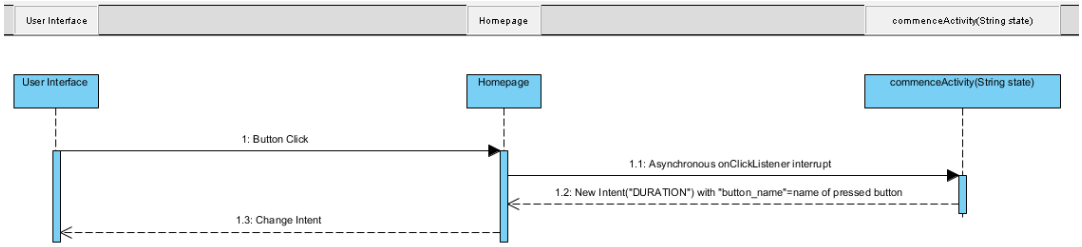


FIGURE C.2: Sequence Diagram of Homepage class, method commenceActivity().

C.2 MainActivity

C.3 Duration

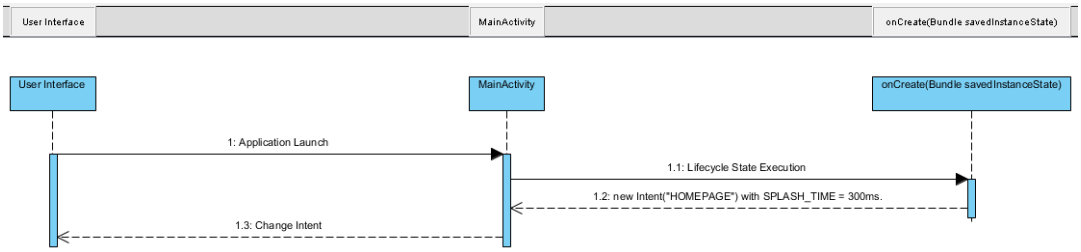


FIGURE C.3: Sequence Diagram of MainActivity class, method onCreate().

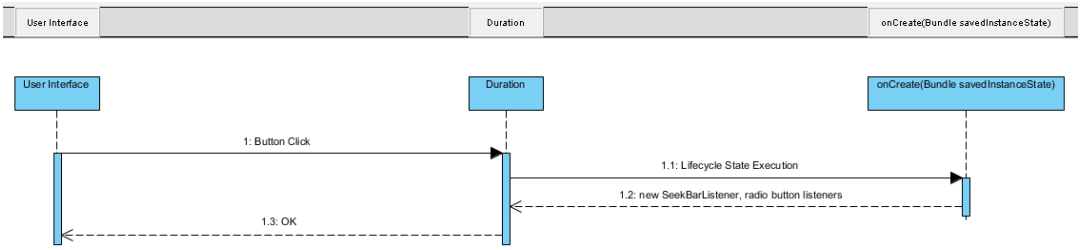


FIGURE C.4: Sequence Diagram of Duration class, method onCreate().

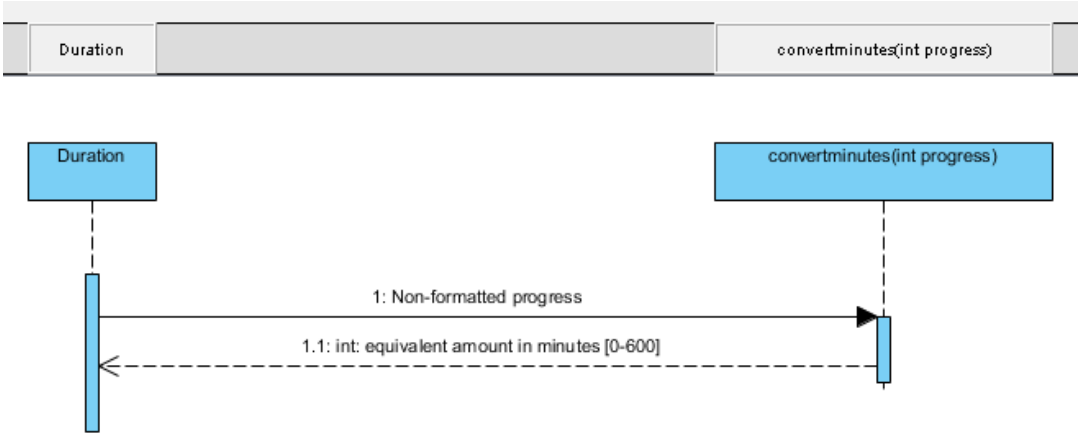


FIGURE C.5: Sequence Diagram of Duration class, method convertMinutes().

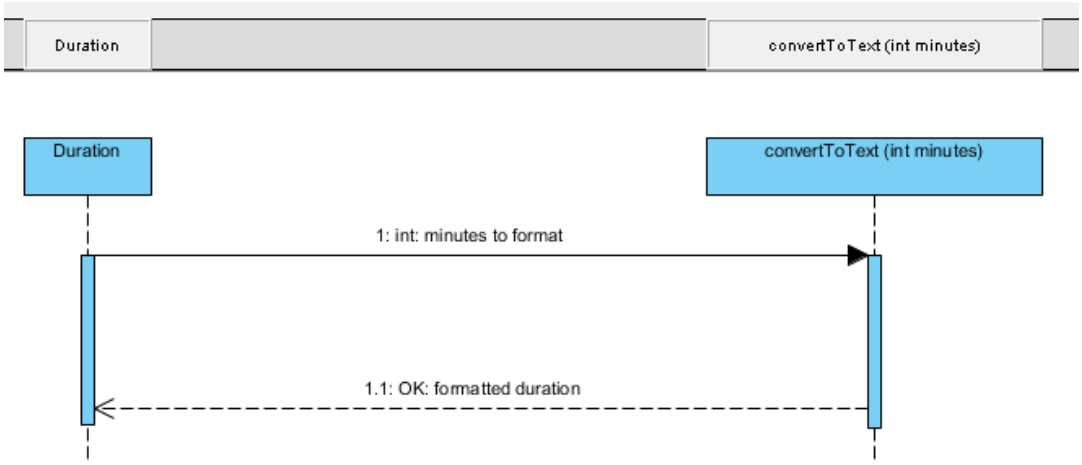


FIGURE C.6: Sequence Diagram of Duration class, method convertToText().

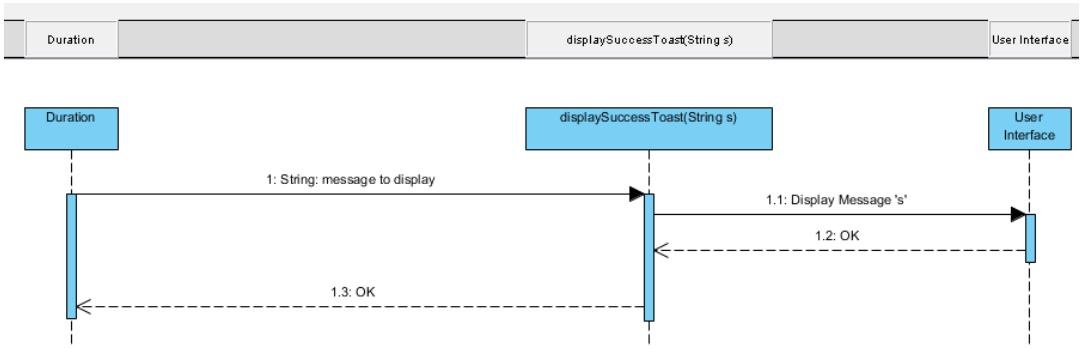


FIGURE C.7: Sequence Diagram of Duration class, method displaySuccessToast().

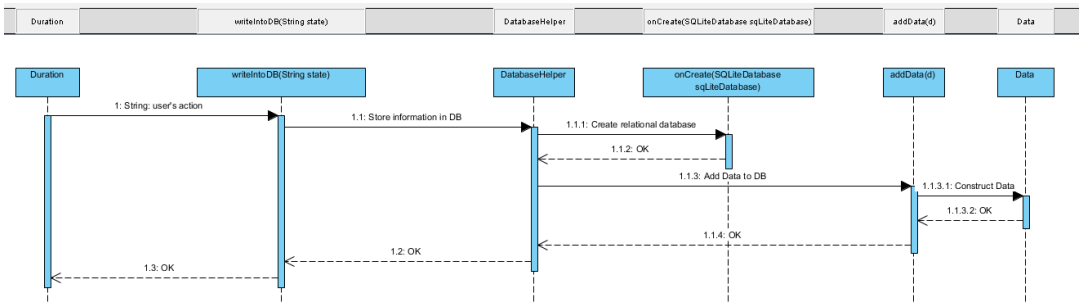


FIGURE C.8: Sequence Diagram of Duration class, method writeIntoDB().

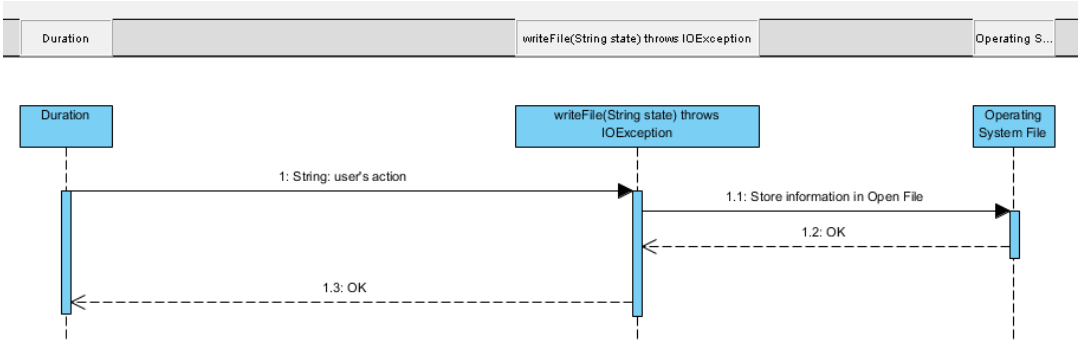


FIGURE C.9: Sequence Diagram of Duration class, method writeFile() on a correct execution.

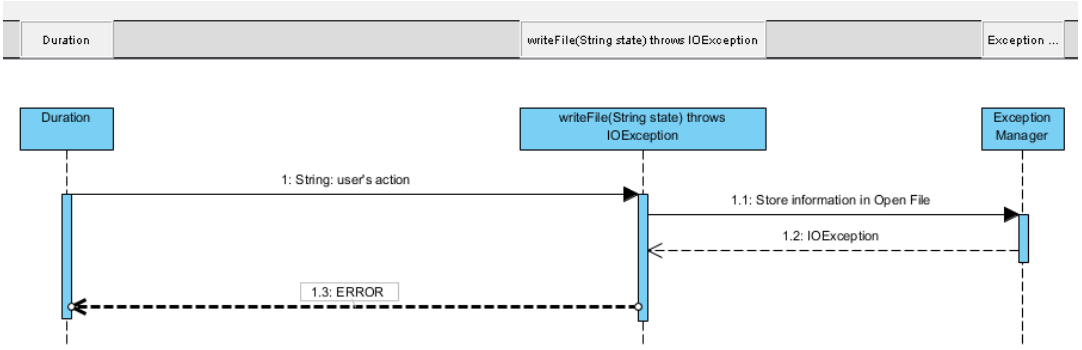


FIGURE C.10: Sequence Diagram of Duration class, method writeFile() on an incorrect execution (Exception).